UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Diseño e implementación de una red de sensores basada en protocolos IoT para monitorización de mercancías.

Máster Universitario en Ingeniería de Telecomunicación

Autor: GÓMEZ MORENO, Rubén

Tutor: LÓPEZ DE VERGARA MÉNDEZ, Jorge E.

FECHA: Febrero, 2020

Diseño e implementación de una red de sensores basada en protocolos IoT para monitorización de mercancías

AUTOR: Rubén Gómez Moreno

TUTOR: Jorge E. López de Vergara Méndez

Dpto. de Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Febrero de 2020

Resumen

Internet de las Cosas se ha convertido en una tecnología que acompaña la vida de todas las personas. Buscamos facilitar las tareas, automatizarlas, y evitar en lo posible los riesgos de los humanos a la hora de llevarlas a cabo, además de hacer el trabajo humano lo más cómodo posible. Cada vez necesitamos tener a nuestro alcance más datos, acerca de cualquier cosa que podamos imaginar, lo que conlleva a implementar tecnología en todas ellas que monitoricen y nos faciliten esos datos.

Este Trabajo de Fin de Máster se centra en la aplicación del Internet de las Cosas para disminuir la cantidad de productos desechados por llegar en mal estado, debido a una mala manipulación en la cadena de transporte. Actualmente se deterioran a diario productos que no pueden ser vendidos, debido a que en su transporte no han viajado en las condiciones adecuadas. El sistema desarrollado en este trabajo trata de disminuir este impacto monitorizando cómo viaja en cada momento el envío realizado, así como de contribuir al desarrollo sostenible.

Entre las dificultades que plantea hacer un sistema para ello, se encuentran que estará en movimiento, serán múltiples envíos a la vez los que deban ser controlados, evaluar las constantes que son más interesantes a tener en cuenta, cómo y dónde analizar los datos recibidos, y lo más importante, que sean presentados una vez analizados de tal forma que tengan utilidad y se pueda acceder a ellos desde cualquier lugar.

Para resolver esto, se plantea un sistema que se divide en en tres grandes bloques: por un lado el sistema que sensoriza la mercancía, midiendo así la temperatura y humedad a la que viaja, y si se producen golpes en la misma; por otro lado, un dispositivo en el contenedor que transporta las mercancías, que recibirá los datos de todos los envíos que contiene, y enviará en tiempo real la localización del contenedor junto con los datos de las medidas recibidas hacia un servidor en la nube. Una vez los datos llegan a la nube, estos son analizados y presentados en un cuadro de mando accesible vía Internet desde cualquier lugar.

Palabras clave

Internet de las cosas, Elasticsearch, Kibana, LTE, wifi, Raspberry, sensores, MQTT, Arduino, macrodatos.

Abstract

Internet Of Things has become a technology in the people's lives. We are looking for easing tasks, automating them, and avoiding as much as possible the risks of humans when carrying them out, and also to make human work as comfortable as possible. In addition, we need to have more and more data at our fingertips, about anything we can imagine, which entails implementing technology in all of them for monitoring and get that information.

This master's final work is focused on the application of the Internet of Things to reduce products wasting, due to poor handling in the transport chain. Currently, there are a large amount of products that cannot be sold, because they have not traveled in the right conditions in their transport. The system developed in this work tries to reduce this impact by monitoring how the shipment has traveled, as well as contributing to sustainable development.

Among the difficulties to make this system, are that it will be in movement, it will be multiple shipments at the same time that must be controlled, evaluate which constants are more interesting to take into account, how and where to analyze the data received, and the most important fact, the data must be presented once analyzed from a way that it would be useful and can be accessed from anywhere.

To solve this, the system proposed can be explained into three large blocks, on the one hand the system that senses the merchandise, measuring the temperature which it travels at, the humidity, and if there are shocks on it. On the other hand, a device inside the container which transports the goods, will receive the data of all the shipments that it contains, and will send in real time the location of the container along with the data of the measurements received towards a server in the cloud. Once the data reaches the cloud, it is analyzed and presented in a dashboard accessible on Internet from anywhere.

Key Words

Internet of things, IoT, Elasticsearch, Kibana, LTE, wifi, Raspberry, sensors, MQTT, Arduino, Big Data.

Agradecimientos

Me gustaría dar las gracias antes de todo a mi pareja, por su apoyo constante animándome a seguir adelante cuando yo estaba a punto de tirar la toalla, no ha sido fácil llegar hasta aquí y siempre me ha animado a superarme y no dejarlo.

Por otro lado, a mis jefes. Durante todo este máster no ha sido una tarea sencilla trabajar a jornada completa y cursar el máster, y ellos me han dado siempre flexibilidad dentro de lo posible para que pudiese seguir con ello, y animándome a realizarlo. Muchas gracias por vuestra comprensión.

A mi tutor, Jorge, porque he tenido que cambiar mil y una tutorías por problemas para cuadrar mi calendario, y jamás me ha respondido con un no, intentando ajustarse a mi disponibilidad para poder ayudarme. Además, siempre ha estado dispuesto a echarme una mano y ha hecho seguimiento constante de mi avance. Gracias por todo.

A mi compañera y amiga Gema, porque aunque no hemos ido de la mano todo el máster, hemos compartido una gran parte de él, y hemos compartido también puesto de trabajo durante ese tiempo, apoyándonos el uno en el otro para sacar esto adelante como fuese posible.

Y por último a mis amigos fuera de la universidad, por soportar mis ya repetitivas charlas sobre lo agobiado que estaba, y de nuevo apoyarme para que no lo dejase, un pensamiento que se ha repetido tantas veces en mi cabeza.

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario

AP	*Access Point,* punto de acceso.
API	*Application Programming Interface,* interfaz de programación de aplicaciones.
AWS	*Amazon Web Services,* servicios web de Amazon.
CoAP	*Constrained Application Protocol,* protocolo de aplicación restringida.
DAFO	Debilidades, Amenazas, Fortalezas, Oportunidades.
GPIO	*General Purpose Input/Output,* entrada/salida de propósito general.
GPS	*Global Positioning System,* sistema de posicionamiento global.
HAT	*Hardware Attached on Top,* hardware conectado en la parte superior.
HTTP	*Hypertext Transfer Protocol,* protocolo de transferencia de hipertexto.
I+D	Investigación y Desarrollo.
IoT	*Internet of Things,* Internet de las cosas.
IP	*Internet Protocol,* protocolo de Internet.
LTE	*Long Term Evolution,* evolución de largo plazo.
M2M	*Machine to Machine*, máquina a máquina.
MQTT	*Message Queue Telemetry Transport,* transporte de cola de mensajes de telemetría.
MTTF	*Mean Time To Failure,* tiempo medio de fallo.
NAT	*Network Address Translation,* traducción de direcciones de red.
ONU	Organización de las Naciones Unidas.
OSI	*Open System Interconnection,* modelo de interconexión de sistemas abiertos.
QoS	*Quality of Service,* calidad de servicio.
REST	*Representational State Transfer,* transferencia de estado representacional.
SBC	*Single Board Computer,* ordenador de placa simple.

SIM	*Susbcriber Identity Module*, módulo de identificación de abonado.
SNMP	*Simple Network Management Protocol,* Protocolo simple de administración de red.
TCP	*Transmission Control Protocol*, protocolo de control de transmisión.
UDP	*User Datagram Protocol,* protocolo de datagramas de uso.
URL	*Uniform Resource Locator*, localizador de recursos uniforme.
VPN	*Virtual Private Network,* red privada virtual.

1 Introducción

1.1 Motivación

Vivimos en un mundo en el que las personas ya están permanentemente conectadas. Pero nos dirigimos un paso más allá, y queremos tener también conectados todos los objetos, y que la tecnología siga avanzando. De esta tendencia, nace IoT (*Internet of Things*), o Internet de las Cosas [1]. Ciudades conectadas, coches conectados, casas conectadas, todo a nuestro alrededor comienza a cobrar vida mediante una conexión a Internet y algo de "inteligencia". Uno de estos entornos, donde cada vez más invierte el sector, es en gestión de flotas, lo que permite a la empresa tener localizados sus vehículos, aportando a la vez seguridad y eficiencia.

Este trabajo, enfocado en este caso de uso del Internet de las Cosas, pretende mejorar los sistemas de gestión de flotas, yendo un paso más allá, y controlando también el estado de la mercancía que transporta cada contenedor. De esta forma, la empresa transportista puede ofrecer garantías de calidad en sus entregas y transportes, a la vez que datos que lo corroboren incorporando estos datos al mundo del *Big Data.* Entre las dificultades que presenta este trabajo, se encuentran las siguientes:

- Dispositivos: optimizar su coste, tamaño y consumo.
- Protocolo de comunicación entre dispositivos: adecuados al caso de uso y a los dispositivos.
- Comunicaciones del dispositivo con Internet.
- Acceso y presentación de los datos.

Para ello, se propone utilizar un SBC (*Single Board Computer*, ordenador de placa simple) como servidor y módulos de bajo coste para sensorizar y transmitir el estado de la mercancía. Esto se realizará mediante un protocolo IoT, como MQTT (*Message Queue Telemetry Transport,* transporte de cola de mensajes de telemetría) o CoAP (*Constrained Application Protocol,* protocolo de aplicación restringida), para poder ahorrar energía y la cantidad de datos a transmitir sobre un enlace inalámbrico como pudiera ser wifi o Bluetooth. Por otro lado, para dar comunicación a Internet y acceso al cliente, se realizará una conexión vía LTE (*Long Term Evolution,* evolución de largo plazo) hacia un servicio *cloud* (en la nube), que será accesible desde cualquier punto con conexión a Internet tanto para el cliente como para la empresa.

1.2 Objetivos

Aproximadamente el 3% de los productos que se suministran a los supermercados, acaban siendo desechados [25]. Entre los principales motivos está que se encuentran en mal estado y no son adecuados para su venta. Muchos de estos productos llegan en mal estado debido a la cadena de transporte, donde si no se manipulan adecuadamente se dañan las condiciones óptimas del producto. El objetivo de este trabajo será, a través de tecnología IoT, monitorizar el envío de mercancías aportando datos de las condiciones en las que se transporta dicho envío, así como su localización. Para ello, se engloban los siguientes subobjetivos:

- Monitorizar las condiciones de temperatura y humedad en el transporte de la mercancía.
- Monitorizar que el envío no sufre golpes o caídas.
- Tener seguimiento GPS (*Global Positioning System,* sistema de posicionamiento global) en tiempo real del envío.
- Tener acceso a los datos del envío vía Internet en un servidor *cloud*.

Estos objetivos será posible llevarlos a cabo mediante sensores, conectados a través de un SBC a Internet contra un servidor *cloud*, que almacena los datos y los muestra en un cuadro de mando. Para validarlo, se desarrollará una maqueta que cubra los aspectos anteriores, visualizando los datos recibidos y las trazas de los mismos. Esto ayudará a la empresa a monitorizar la cadena de transporte, pudiendo comprobar si una mercancía en mal estado puede deberse a unas malas condiciones de transporte, además de participar como medida de desarrollo sostenible, puesto que reducimos la cantidad de alimentos que se desperdician. Esto ayuda a cumplir los objetivos de la ONU (Organización de las Naciones Unidas), cuyas metas del objetivo número 12 de producción y consumo responsables, engloban entre ellas "de aquí a 2030, reducir a la mitad el desperdicio de alimentos per cápita mundial en la venta al por menor y a nivel de los consumidores y reducir las pérdidas de alimentos en las cadenas de producción y suministro, incluidas las pérdidas posteriores a la cosecha" [26].

1.3 Fases de realización

La realización de este trabajo abarca 15 meses aproximadamente, en los que se dividen las tareas según el siguiente diagrama de Gantt (Figura 1-1):

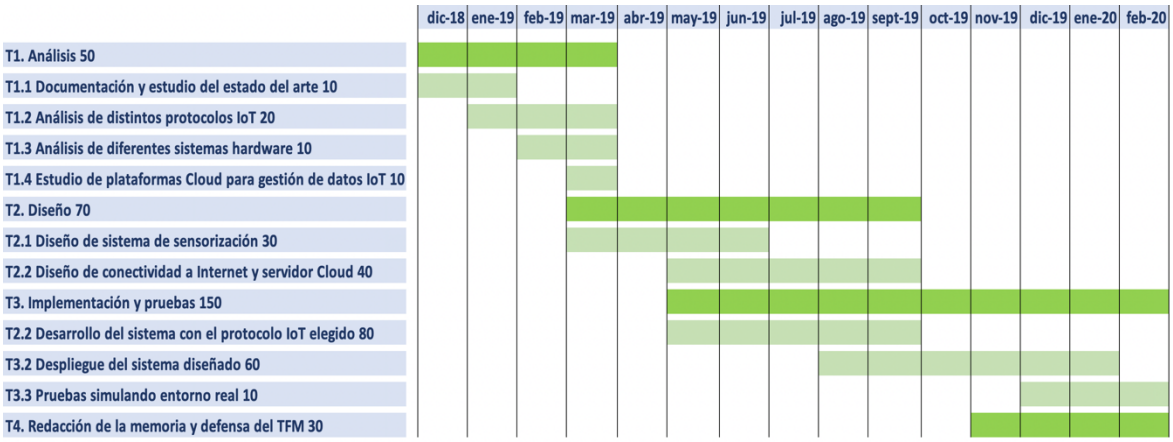

Figura 1-1: Diagrama de Gantt

Este trabajo se ha realizado según las fases descritas en los siguientes apartados:

1.3.1 Documentación

En esta fase del trabajo, se ha realizado un trabajo de investigación y documentación acerca del estado del arte sobre los casos de uso IoT. Esto ha permitido posicionarse sobre el estado actual de nuevos desarrollos IoT, la demanda de nuevos casos de uso para esta tecnología, y problemas que pueden ser resueltos con dispositivos IoT actualmente.

Posteriormente, se ha documentado el caso de uso a desarrollar, así como los requisitos y desarrollos necesarios del mismo, pasando a la fase de análisis.

1.3.2 Análisis

En esta fase se analizó el problema propuesto en este caso de uso y las posibles soluciones. Para este caso de uso, monitorización de envíos en mercancías, es necesario analizar cómo y dónde actúa el IoT, es decir, dónde estaría la "inteligencia" de la mercancía transportada. Así, en este trabajo, se propone un conjunto de dispositivos que monitorizan conjuntamente mercancía y contenedor de transporte en tiempo real con conexión a Internet.

Por otro lado, es importante elegir protocolos adecuados. Para ello, bajo estudio y realizando pruebas, a nivel de enlace se opta por usar wifi entre el dispositivo en la mercancía y el SBC principal del contenedor, LTE para la conexión del SBC principal con la nube; en el nivel de aplicación, tras analizar CoAP o MQTT que son los más usados actualmente, se opta por MQTT, para comunicar los datos de las medidas los sensores del envío con el SBC principal.

Respecto a la sensorización, interesante para este caso de uso son datos como la temperatura a la que se transporta la mercancía, la humedad a la que va sometida, y que no se produzcan daños por golpes a lo largo del transporte. Por ello, serán estas tres variables las que se sensorizan en este trabajo.

1.3.3 Diseño e implementación

En esta fase se implementan tanto los sensores como las comunicaciones de los mismos. Una vez implementados los sensores, se programa en el módulo nodeMCU la comunicación MQTT de los datos hacia el SBC principal usando un bróker MQTT. En el SBC principal, se programa el envío de los datos recibidos hacia la nube por LTE, y son guardados, organizados y presentados con la ayuda de Elasticsearch y una interfaz de presentación de un cuadro de mando.

1.3.4 Validación y pruebas

Se realizan pruebas parciales del trabajo, probando la funcionalidad de cada elemento por separado, así como de cada conectividad implementada y su rendimiento. Una vez comprobado el correcto funcionamiento, se realizan también pruebas de consumo energético, y se prueba el sistema al completo validando así la funcionalidad y objetivo del mismo.

1.3.5 Redacción

En esta fase se ha redactado esta memoria del trabajo. En ella se detallan las diferentes fases de estado del arte, análisis, diseño, implementación, validación y pruebas del trabajo, estructurando el documento conforme a la realización del mismo. En el siguiente apartado se puede observar la estructura del documento.

1.4 Estructura de la memoria

Esta memoria sigue una estructura clásica, con los siguientes capítulos:

1. *Introducción*: se introduce el trabajo de fin de máster, así como el mundo IoT y la motivación del desarrollo de este sistema.

2. *Estado del Arte*: en este capítulo se describe el estado del arte actual en IoT o Internet de las Cosas, así como de las aplicaciones en el caso de uso planteado, la monitorización del transporte de mercancías.

3. *Análisis*: en este capítulo se expone el análisis realizado del caso de uso, la problemática asociada, y cómo se plantea resolverlo a través de este trabajo. Para ello se analiza una propuesta de solución, los requisitos de la misma, y las decisiones tomadas para su posterior implementación.

4. *Diseño e implementación*: en este capítulo se explica el diseño elegido, los componentes del mismo, así como la implementación que se lleva a cabo para este trabajo, cubriendo así los problemas planteados y ofreciendo una solución optimizada para el caso de uso.

5. *Validación y pruebas*: en este capítulo se detallan las pruebas realizadas una vez implementado el diseño, con el fin de validar que los requisitos del sistema han quedado cubiertos, así como que funciona correctamente. Con ello se valida el trabajo realizado.

6. *Conclusiones*: En este capítulo se desarrollan las conclusiones obtenidas del trabajo realizado, y se proponen nuevas líneas de trabajo futuro que podrían mejorar el sistema implementado.

2 Estado del arte

2.1 Introducción

La tendencia actual es la conexión a Internet de cualquier cosa que podamos imaginar, desde una bombilla, a un juguete, un electrodoméstico o, el chip de un animal para su control. En este capítulo, se trata el estado del arte actual del mundo IoT, abordándolo tanto desde la perspectiva de los propios casos de uso de esta tecnología, como desde la perspectiva del *Big Data* que el mundo IoT genera, algo muy importante y que cada día las empresas valoran más, pero que a la vez necesita de una adecuada gestión.

A continuación, se describe Internet de las cosas, así como sus aplicaciones en diferentes ámbitos, casos de uso en el estado del arte actualmente. Posteriormente se explican los protocolos IoT más conocidos y por último, la gestión y el análisis de los datos generados por el mundo IoT.

2.2 IoT

IoT, *Internet of Things* o Internet de las cosas, se define como la interconexión digital de personas, animales y cosas (electrodomésticos, coches, etc.) con Internet [38]. Así, como se mencionaba en el apartado anterior, actualmente se dota a cualquier objeto que usamos en nuestro día a día de conexión a Internet, facilitando la vida de las personas. De esta forma, podemos automatizar procesos, tener al alcance de nuestra mano el control de dichos objetos, y obtener datos continuos acerca del mismo.

El mundo IoT tiene un abanico sin límites de campos a los que se puede aplicar, en diferentes sectores como:

- Industria: seguimiento de flotas, robotización de procesos, sensorizaciones, automatismos, etcétera.
- Comercio: control de stock, seguimiento de mercancías, etcétera.
- Sanidad: telemedicina, monitorización de constantes vitales de forma inalámbrica, telecirugía, etcétera.
- Geriatría: localización de personas, control de constantes vitales, botones del pánico, etcétera.
- Hogar: electrodomésticos conectados, iluminación inteligente, climatización, etcétera.
- Transporte: coches autónomos, coches conectados, etcétera.

Según IoT Analytics [2], la distribución en 2018 por casos de uso se distribuye según la siguiente figura (Figura 2-1):

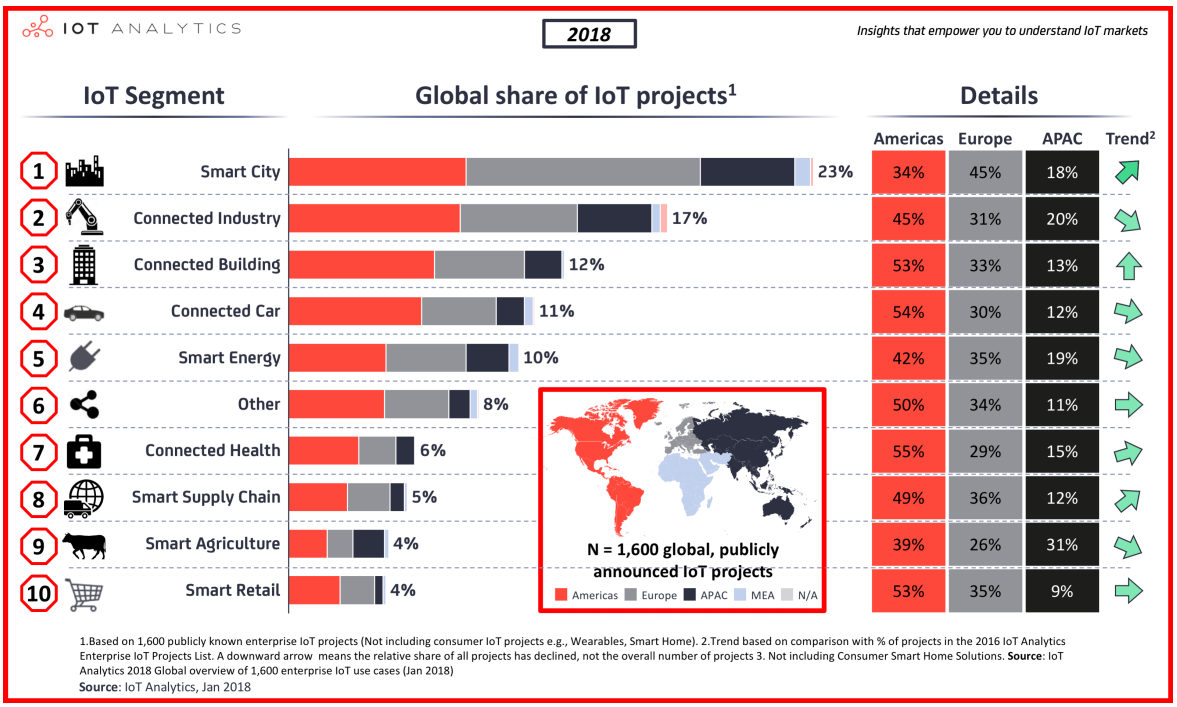

Figura 2-1: Crecimiento IoT [2]

Se puede observar que los segmentos con más crecimiento actualmente corresponden también con los más desarrollados a día de hoy, pero hay un claro crecimiento en todos los sectores. A continuación, se detalla el estado del arte de algunos de estos casos de uso.

2.2.1 Casos de uso

Uno de los casos de uso que más se está extendiendo es la domótica. Dotamos de conectividad a la iluminación, el termostato, los electrodomésticos, e incluso disponemos de asistentes virtuales de voz que nos permiten gestionar todo esto con solo hablar.

Pero hay otros casos de uso que no van solo a la comodidad, sino que pueden llegar a salvar una vida o a ahorrar cantidades de dinero elevadas a las empresas. Hablamos, por ejemplo, de dispositivos que monitorizan de forma inalámbrica las constantes vitales de un paciente permitiendo así al médico conocer en todo momento el estado de este, y alertarle en caso de que algo no vaya bien. A continuación, se reseñan algunos de los proyectos más innovadores actualmente en el mundo IoT.

- **Bus autónomo EZ10**

Se trata de un vehículo cien por cien eléctrico y autónomo, diseñado por la empresa EasyMile, con capacidad para 12 pasajeros y que realiza un recorrido de 3,8 Km en el campus de Cantoblanco, de la Universidad Autónoma de Madrid [3]. Es el primer vehículo en España de estas características.

Este vehículo es autónomo gracias a la dotación de sensores e inteligencia que le permite saber dónde se encuentra, ubicarse en el recorrido, frenar si detecta obstáculos, o parar cuando llega a sus destinos establecidos. Se puede observar en la siguiente imagen (Figura 2-2) el bus en el campus.

Figura 2-2: Bus EZ10 [3]

- **Samsara AI Dash Cam**

La compañía Samsara [4] ha desarrollado una *Dash Cam* que se puede poner en cualquier vehículo, y la cual mediante inteligencia artificial identifica peligros durante la conducción.

Detecta, por ejemplo, si la distancia entre un vehículo y otro es demasiado pequeña, o si hay riesgo de choque por deceleración del vehículo precedente, señales de tráfico, etcétera. Ante un riesgo, una voz alerta al conductor de este, y a la vez realiza una fotografía que sube a un servidor *cloud*. Desde un cuadro de mando *online*, la empresa o el conductor puede ver la puntuación del conductor o los reportes de alerta entre otros datos. También monitoriza el interior del vehículo detectando si ha sido debido a una distracción del conductor y tomando una foto de este. En la siguiente figura (Figura 2-3) se puede ver el dispositivo.

Figura 2-3: Al Dash Cam [4]

- **Smarter FridgeCam**

De la empresa Smarter, encontramos la *FridgeCam [5]*, un dispositivo que se puede instalar en cualquier frigorífico y que se trata de una cámara que toma una foto cada vez que cerramos el frigorífico. Posteriormente, esta foto será siempre accesible desde una aplicación que no sólo se limita a mostrar la fotografía, sino que nos permite ver un listado de los productos en nuestro frigorífico, ver una lista de productos agotados pendientes para la lista de la compra e incluso ver qué productos están a punto de caducar.

Además, es compatible con los asistentes de voz *Siri y Alexa*, lo que permite consultar por voz qué productos van a caducar próximamente o si hay existencias de un alimento en tu frigorífico sólo con la voz. Por otro lado, gracias a acuerdos con cadenas de supermercados, puedes pedir que te haga la compra de los productos que se han agotado o están a punto de agotarse. Todo esto es posible gracias al IoT, ya que este dispositivo se conecta por wifi a Internet. Debido a su reducido tamaño (Figura 2-4) no ocupa prácticamente espacio dentro del frigorífico.

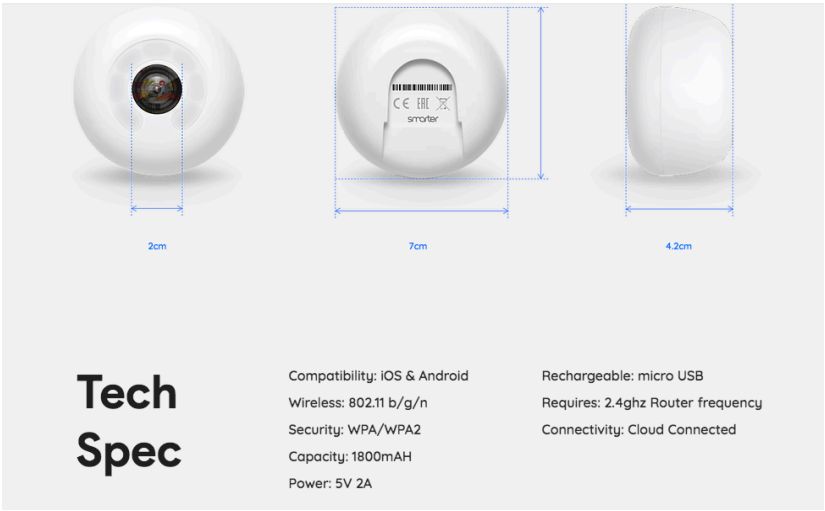

Figura 2-4: Smart FridgeCam [5]

2.3 Protocolos IoT

Como cualquier protocolo de red, los protocolos IoT también tienen clasificación dentro del modelo OSI (*Open System Interconnection,* modelo de interconexión de sistemas abiertos) según la finalidad del protocolo. En la siguiente figura (Figura 2-5) se puede ver el modelo OSI:

Figura 2-5: Modelo OSI

Comenzando por los niveles físico y de enlace, en IoT existen múltiples opciones. Algunas de ellas son BLE [27], Sigfox [28], LoRa [29], wifi [30], LTE [31], NB [32], etcétera. Entre los más usados actualmente destacan BLE o Bluetooth de baja energía, wifi, NB o banda estrecha, o LTE. La razón de esto es que son los más extendidos y por tanto conocidos, por lo que es más fácil encontrar documentación al respecto y equipos compatibles.

A nivel de red, se utiliza el protocolo de Internet o IP (*Internet Protocol,* protocolo de Internet) como es habitual. Si pasamos al nivel de transporte, ambos protocolos son utilizados, tanto UDP (*User Datagram Protocol,* protocolo de datagramas de uso) como TCP (*Transmission Control Protocol*, protocolo de control de transmisión) en función del diseño y del caso de uso. En este trabajo, debido a que se elige MQTT como protocolo de la capa

de aplicación, se usará TCP en la capa de transporte. A nivel de sesión y presentación, tampoco destaca nada específico para IoT.

En la capa de aplicación, encontramos diferentes protocolos que se usan en el mundo IoT, como, por ejemplo, SNMP (*Simple Network Management Protocol,* Protocolo simple de administración de red), CoAP, MQTT o API (*Application Programming Interface,* interfaz de programación de aplicaciones) REST (*Representational State Transfer,* transferencia de estado representacional). Para este trabajo, se analizan los dos protocolos más utilizados, CoAP y MQTT [33].

2.3.1 CoAP

El protocolo CoAP es un protocolo software del nivel de aplicación, definido en la RFC 7252 [7] como *un protocolo de transferencia web específico para ser usado con nodos y redes acotadas en IoT.* Este protocolo está basado en el modelo REST, en el cual los recursos del servidor son alcanzables a través de una URL (*Uniform Resource Locator*, localizador de recursos uniforme) y el cliente accede a ellos usando los métodos PUT, POST, GET y DELETE. El protocolo CoAP se caracteriza por ser interoperable con HTTP, permite intercambio asíncrono de mensajes, y se transmite por UDP.

El diagrama de diálogo de CoAP es bastante simple. En un caso de uso como el de este trabajo, se enviaría un mensaje de CON y GET para obtener el dato del sensor y, el SBC contestaría con un ACK y el dato, como se observa en la siguiente imagen (Figura 2-6):

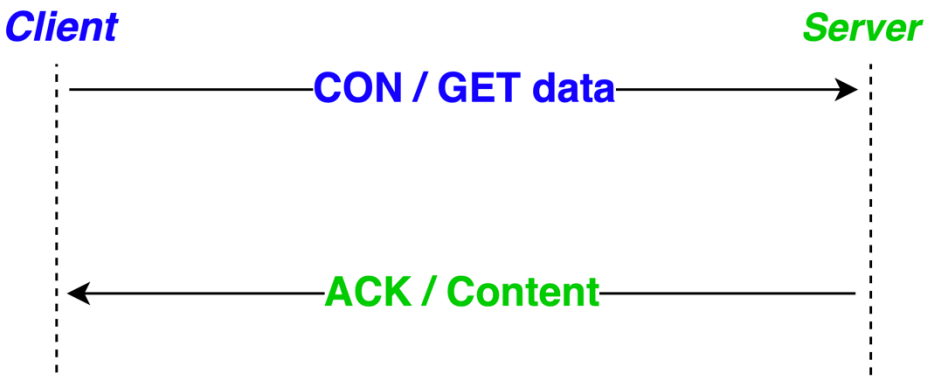

Figura 2-6: Diálogo CoAP

En cuanto al formato de los mensajes, sigue la siguiente estructura binaria (Figura 2-7):

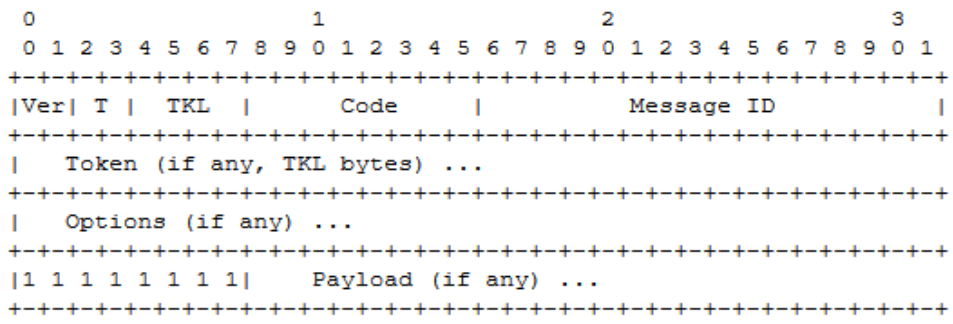

Figura 2-7: Paquete CoAP [7]

En cuanto a la arquitectura, CoAP está basado en una arquitectura REST, extendiendo el HTTP normal, pero con limitación de recursos.

2.3.2 MQTT

El protocolo MQTT es un protocolo software a nivel de aplicación, desarrollado por el Dr. Andy Stanford-Clark de la empresa IBM y Arlen Nipper, de la empresa Arcom. La versión más actual, v5.0, es un estándar de OASIS [8]. Es un protocolo con modelo *publish/suscribe,* muy ligero, pensado para redes con poco ancho de banda, alta latencia o poco fiables. A diferencia de HTTP, el protocolo MQTT es bidireccional, permite que sea asíncrono, no está limitado a una topología uno a uno, y es un protocolo mucho más ligero, por ello en muchos desarrollos de IoT se usa MQTT y no HTTP. Un ejemplo de diálogo estándar en un caso de uso de MQTT podría ser como en el siguiente esquema (Figura 2-8):

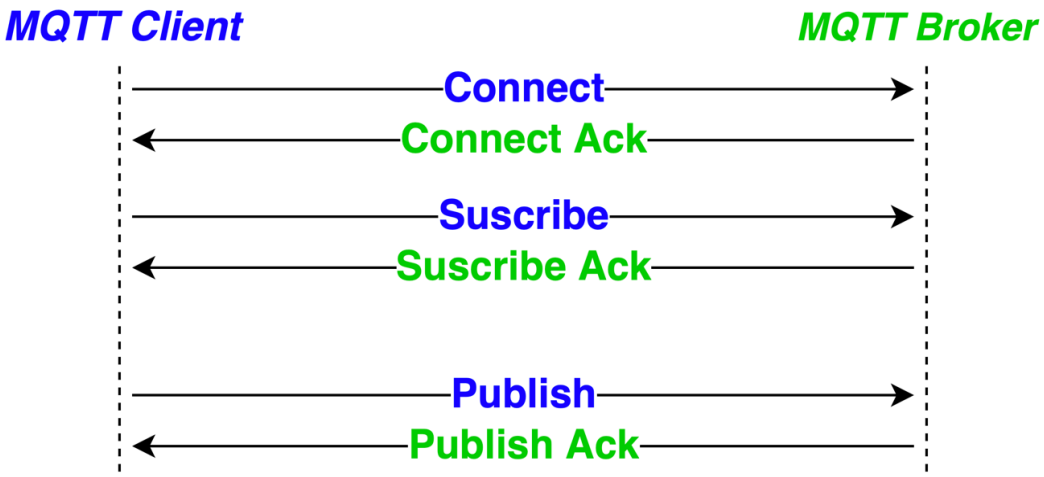

Figura 2-8: Diálogo MQTT

En el mismo se puede observar que tras un mensaje *connect* con el bróker, se envía un *subscribe* hacia un *topic*, de tal forma que cuando haya un nuevo dato disponible en ese *topic*, se recibirá. Además, el bróker puede hacer una petición de *publish* y publicar un dato concreto en el dispositivo final.

En cuanto a la estructura de un paquete del protocolo MQTT, sigue el siguiente formato (Figura 2-9):

Fixed Header, present in all MQTT Control Packets
Variable Header, present in some MQTT Control Packets
Payload, present in some MQTT Control Packets

Figura 2-9: Paquete MQTT [8]

A diferencia de CoAP, el protocolo MQTT funciona por TCP y no por UDP. MQTT permite configurar 3 niveles de QoS (*Quality of Service,* calidad de servicio) y de esta forma, garantizar la entrega del paquete, como máximo una vez, al menos una vez o exactamente una vez. Las redes en las que se usa MQTT tienen un bróker, y varios *publisher/subscriber* con los que se comunica. Un modelo de arquitectura MQTT habitual tendría el siguiente esquema (Figura 2-10):

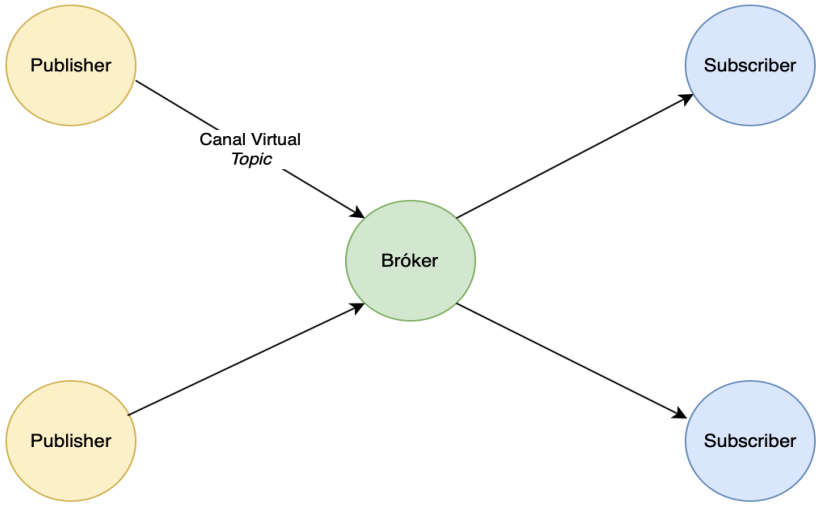

Figura 2-10: Red MQTT típica

Esto permite que el SBC principal que hace de bróker, actúe a la vez de agregador tratando todos los datos que recibe según se le haya programado.

2.4 Big Data orientado a IoT y análisis en la nube

En el mundo IoT se generan cantidades ingentes de datos que para ser útiles necesitan ser procesados, poder ser analizados, y que se presenten correctamente. Es aquí donde enlaza con el mundo *Big Data*, de tal forma que los datos generados se procesen y se almacenen, y sean accesibles para poder ser tratados.

Una de las líneas de *Big Data* con más éxito en la actualidad, es el tratamiento y almacenamiento en la nube o *cloud,* ya que esto permite que sea accesible desde Internet, es decir, desde cualquier punto, no requiere recursos propios, y además garantiza seguridad. Existen diferentes sistemas de análisis [18] con diferentes ventajas, reflejados en la siguiente tabla (Tabla 2-1):

Tabla 2-1: Sistemas de análisis *Big Data*

Tipos/nivel de análisis	Uso específico	Ventajas/categoría
Real time	Para analizar grandes cantidades de datos generadas por los sensores	Clasificadores de procesamiento en paralelo utilizando bases de datos tradicionales.
Offline	Para aplicaciones donde el tiempo de respuesta no es un requisito	Reduce el coste de conversión de formato de los datos. Adquisición de datos eficiente.
Memory level	Para escenarios donde el volumen total de datos a procesar es mayor que la memoria del clasificador	Tiempo real
Business intelligence level	Cuando la escala de datos supera el nivel de memoria	Tanto en tiempo real como *offline*
Massive level	Cuando la escala de datos supera totalmente la capacidad de los productos de inteligencia de negocio y las bases de datos tradicionales	Mayormente *offline*

El análisis en la nube es uno de los pilares fundamentales del *Big Data*. A día de hoy, resulta impensable con la cantidad de datos que se mueven y se analizan, y en constante

crecimiento, que puedan ser almacenados y analizados de forma local. Además, que todo esto se realice en la nube, aporta numerables ventajas como:

- Seguridad: habitualmente son sistemas redundados y accesibles por múltiples rutas, además de tener la información cifrada.

- Ahorro de costes: ahorra tener múltiples sistemas de almacenamiento local conectados en lugar de acceder desde múltiples lugares a un mismo sitio.

- Capacidad de procesamiento: la capacidad de procesamiento en la nube es flexible y mucho mayor de lo que sería un equipo en local.

- Acceso: la información está al alcance de cualquier sistema que tenga salida a Internet.

Dentro de las herramientas que se usan para análisis de datos, están entre otras, Apache Hadoop [34], MongoDB [35] o Elasticsearch [13]. Cualquiera de ellas es válida, pero según el sistema a diseñar y los elementos que lo compongan puede ser más sencillo o adecuado una herramienta u otra.

2.5 Conclusiones

Si el crecimiento en dispositivos con tecnología IoT de los últimos años ha sido notable en cualquier ámbito, para los que siguen no será menos. Se espera que se superen los 50 miles de millones de dispositivos IoT en 2020 [36], y cada vez son más los dispositivos preparados para conectarse a Internet.

Esta tendencia hace que sean cada vez más las empresas que apuestan por el IoT, y que el estado del arte en este ámbito cambie constantemente. Además, todo esto genera cantidades de datos que crecen constantemente y que para ser de utilidad deben ser analizados y procesados correctamente. Así, el IoT es ahora mismo una tecnología presente en la vida de las personas, y que ha llegado para quedarse.

Una vez visto el estado del arte en IoT, ejemplos de casos de uso, los protocolos IoT que se usan y la importancia del *Big Data* en el mundo del Internet de las cosas, en el siguiente capítulo se analiza el caso de uso concreto a realizar en este trabajo de fin de máster.

3 Análisis

3.1 Introducción

El seguimiento de flotas en empresas de transporte es un pilar fundamental. Necesitan conocer en todo momento donde se encuentran sus repartidores, y por consiguiente sus envíos. Actualmente, la mayor parte de las empresas de transporte únicamente cuenta con seguimiento GPS de sus contenedores, con el fin de saber en qué lugar se encuentran, y con seguimiento de los envíos a través de los terminales que usan los transportistas, donde marcarán si se ha entregado o si ha surgido alguna incidencia.

Pero sería interesante dar un paso más, y llevar el IoT a este ámbito. Así, en este trabajo, se monitoriza no sólo la posición del contenedor, sino el estado en el que ha viajado la mercancía.

En este capítulo se analizan los casos de uso de este sistema, los requisitos que tiene, un posible caso de negocio y por último las conclusiones obtenidas tras el análisis.

3.2 Casos de uso

Para llevar a cabo este trabajo, se han considerado los casos de uso de la siguiente figura (Figura 3-1):

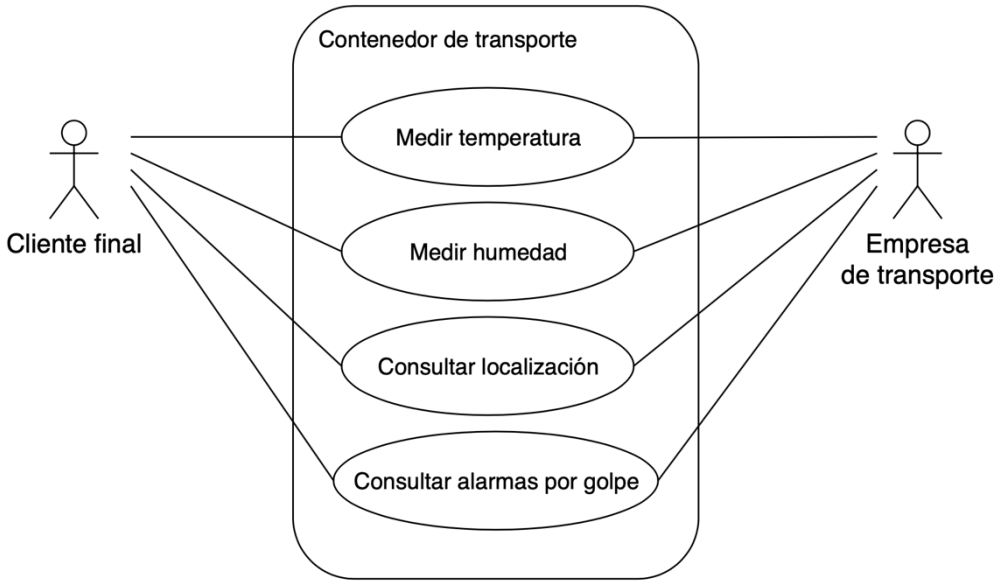

Figura 3-1: Casos de uso

A continuación, se detallan los elementos contenidos en la figura anterior:

- Actores:
    - Empresa de transporte: empresa encargada del transporte de la mercancía, y por tanto responsable del contenedor.
    - Cliente final: propietario que espera la mercancía en el destino o, que ha enviado ésta en origen.
- Casos de uso:
    - Medir temperatura: se debe medir la temperatura a la que se encuentra la mercancía.
    - Medir humedad: se debe medir la humedad a la que se encuentra la mercancía.
    - Consultar la localización: ambos actores podrán conocer en todo momento la localización del contenedor en el que viaja la mercancía.
    - Consultar alarmas por golpes: ambos actores podrán conocer las alarmas detectadas por golpes de la mercancía.

3.3 Requisitos

A pesar de que se han realizado prototipos que introducen el IoT en mercancías, como sensores desechables para los alimentos, o maletas conectadas, lo cierto es que es un ámbito en el que hay actualmente pocos dispositivos y proyectos que se hayan llevado a producción.

En este trabajo, se trata de introducir el IoT en mercancías de una forma sencilla y de bajo coste, monitorizando el estado en el que se encuentra la mercancía en la cadena de transporte. Entre los requisitos en este ámbito podemos encontrar los siguientes:

- Es importante saber las condiciones de temperatura y humedad en las que se transporta una mercancía, ya que una temperatura extremadamente caliente o fría, o un porcentaje muy alto de humedad puede hacer que se estropee.
    - ⇒ La maqueta implementada deberá monitorizar la temperatura en ºC de la mercancía en tiempo real.
    - ⇒ La maqueta implementada deberá monitorizar la humedad en % a la que se encuentra la mercancía en tiempo real.

⇒ Ambos datos deberán ser guardados en un histórico y se deberá poder acceder a ellos desde Internet.

- Si una mercancía sufre un golpe, es probable que resulte dañada o ya no sea apta para su venta.

  ⇒ La maqueta implementada deberá enviar una alarma en el caso de que la mercancía sufra un movimiento brusco.

  ⇒ El estado de alarma deberá ser monitorizado en tiempo real.

  ⇒ Las alarmas deberán ser guardadas en un histórico que posteriormente será accesible desde Internet.

- Es necesario que los datos registrados por los sensores sean almacenados y tratados, para poder presentarse al usuario final en una interfaz gráfica.

  ⇒ La maqueta implementada deberá enviar los datos de los sensores mediante MQTT al bróker MQTT.

  ⇒ El bróker MQTT de la maqueta deberá encargarse de indexar los datos en Elasticsearch a través de Internet.

  ⇒ El SBC que hará de bróker MQTT debe contar con conexión a Internet por LTE.

  ⇒ El SBC que hará de bróker MQTT debe poder comunicarse correctamente con el servidor *cloud* de Elasticsearch para enviarle los datos recibidos.

- Es muy importante tener localizado el contenedor en el que se transporta la mercancía en tiempo real, permitiendo así en caso de que haya un problema con ésta, localizar el punto en el que se ha producido.

  ⇒ La maqueta implementada deberá permitir la localización en tiempo real del contenedor de transporte.

  ⇒ La localización deberá ser mediante GPS.

  ⇒ El SBC principal (bróker MQTT) deberá enviar junto con los datos de los sensores la localización GPS obtenida en tiempo real a través de Internet hacia la nube.

- Todos los datos recogidos necesitan ser tratados y presentados en un cuadro de mando entendible, y que resulte de utilidad.

  ⇒ Este trabajo deberá implementar un cuadro de mando en una interfaz gráfica.

⇒ El cuadro de mando deberá ser accesible por Internet.

3.4 Análisis caso de negocio

En este trabajo, se ha analizado un posible caso de negocio sobre la maqueta implementada para este caso de uso.

En este caso de negocio, se ha tenido en cuenta que:

- El sistema, una vez vendido al cliente final, será mantenido por éste.
- Coste inicial.
- Costes de componentes.
- Costes de reemplazo.
- Costes de formación a personal de mantenimiento.
- Costes de implementación.
- Desgaste de los componentes.
- Se ha supuesto una flota de 1.000 camiones/contenedores, con 25 mercancías o palés en su interior.
- Se han calculado costes tanto con los componentes reales de esta maqueta, como con la aproximación de los costes reales que tendría el cliente final.

En el Anexo A se detalla el presupuesto del mismo. El coste por sistema completo para equipar un contenedor sería de 548€, lo que implica un coste inicial por equipar la flota entera de 553.850 €.

Además de estos costes, habría que añadir un coste de desarrollo inicial de 1.600€ estimados a los que se añade un coste de I+D (Investigación y Desarrollo) de 6.000€ adicionales. Estos costes están estimados de acuerdo con el XIX Convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos [9], cuya tabla salarial nos indica un coste por hora aproximado de 20€ teniendo en cuenta el sobrecoste del 33% para la empresa sobre el salario bruto del empleado.

Por otro lado, es importante tener en cuenta el coste de mantenimiento en periodos posteriores, ya que, aunque será siempre inferior al coste inicial de implementación, el sistema necesita un mantenimiento continuo. En este mantenimiento se valoran costes de

reemplazo de componentes por avería, actualizaciones del dispositivo, reemplazo de batería, así como la formación al personal encargado de la sustitución de la misma, ya que la batería podrá ser cambiada de forma sencilla sin requerir de un técnico para ello.

Con todo ello, se estima un coste anual de 53.409 € anuales, entre los que se incluye el salario del técnico encargado de mantener la red de dispositivos instalados en la flota, que no tendría por qué estar dedicado exclusivamente a este proyecto.

Si consideramos el posible ahorro estimado del 33% de las pérdidas de una empresa por mercancías en mal estado, en una escala de ingresos de millones de € mensuales, en la que el ahorro mensual de un tercio de las pérdidas supera el millón de euros, tendríamos un punto de *break even* (Figura 3-2) desde el primer mes, ya que ni el desembolso inicial ni el coste de mantenimiento superan apenas el medio millón de euros.

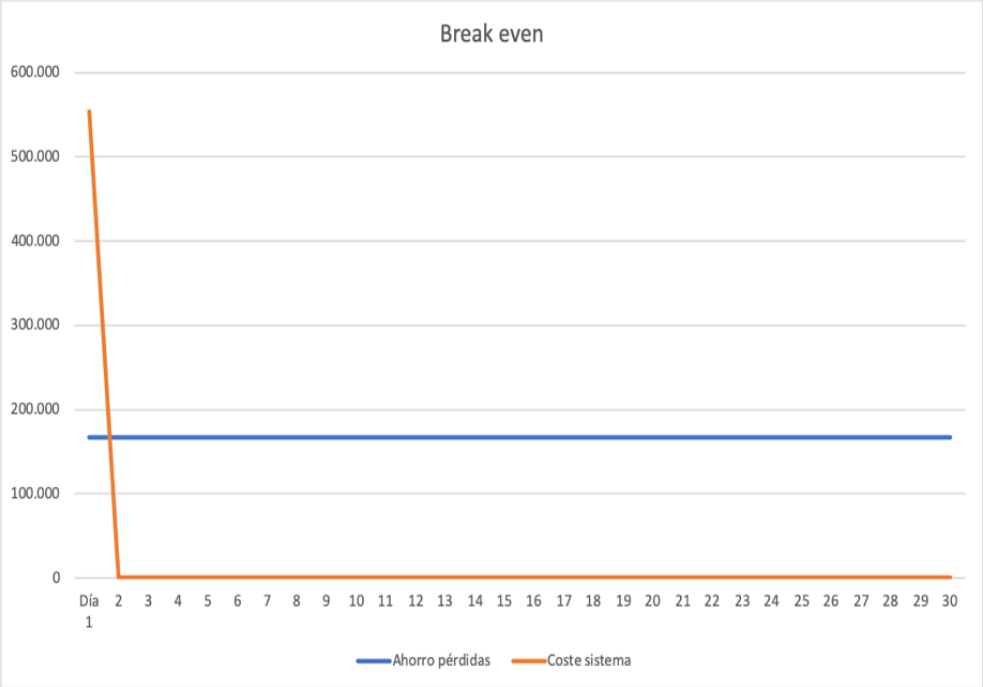

Figura 3-2: Punto de *break even*

Respecto al reemplazo de componentes, según los diferentes *datasheets* analizados para cada uno de ellos, encontramos los siguientes tiempos medios de reemplazo:

Tabla 3-1: MTTF (*Mean Time To Failure*, tiempo medio de fallo) de los componentes

Componente	Vida estimada
DHT11	1 año[1]
mpu6050	1 año
Tarjeta de memoria	6 meses
nodeMCU	5 años
Raspberry	3 años
Batería	5 años

De nuevo, las expectativas para años posteriores son de reducción de pérdidas en los ingresos de la empresa por mercancías dañadas en la cadena de transporte o transportadas en malas condiciones. Si analizamos el DAFO (Debilidades, Amenazas, Fortalezas, Oportunidades) de este caso de negocio, nos queda la siguiente tabla:

Tabla 3-2: DAFO

DEBILIDADES • Manipulable. • Necesita cobertura móvil.	**AMENAZAS** • Competencia. • Evolución de la tecnología.
FORTALEZAS • Rápido *break even*. • Económico. • Disminuye las pérdidas.	**OPORTUNIDADES** • Grandes empresas interesadas en el sistema. • Explotación comercial de los datos.

[1] Aunque el tiempo de vida medio que nos dan los datasheets es de 1 año para el DHT11, la experiencia en este trabajo nos hace suponer que es menor, ya que durante el año de trabajo ha habido que utilizar por fallo tres sensores diferentes.

Entre las debilidades encontramos:

- Manipulable: el dispositivo tal y como se ha implementado en la maqueta, sería manipulable. Es por ello que, en un diseño final para producción, deberá ir en una caja rugerizada que lo proteja de posibles manipulaciones o deterioro por golpes, convirtiéndose así en fortaleza.

- Necesita cobertura móvil: este sistema, para poder transmitir datos, necesita de cobertura móvil. Al estar ideado para transportes por tierra, no debería tener problemas. No obstante, esto es por otro lado una fortaleza, ya que puede trabajar en caso de que no hubiese cobertura LTE con 3G/2G en sus diferentes bandas de frecuencia.

Entre las fortalezas:

- Rápido *break even*: en el primer mes desde su compra, la empresa estaría reduciendo sus pérdidas y por tanto incrementando sus beneficios. El coste inicial no alcanza el medio millón de euros, mientras que la reducción de pérdidas supera el millón de euros.

- Económico: se trata de dispositivos de bajo coste tanto de implementación como de mantenimiento.

- Disminuye las pérdidas: disminuye las pérdidas de la empresa al poder monitorizar cómo se transporta la mercancía.

En las amenazas:

- Competencia: otras empresas podrían interesarse por este proyecto y fabricar sus propios dispositivos, o estar vendiendo dispositivos similares actualmente. Esto a la vez es una oportunidad de negocio, ya que si hay competencia quiere decir que hay clientes interesados en implementarlo en sus negocios.

- Evolución de la tecnología: dado el actual ritmo de avance de la tecnología es posible que quedase obsoleto debido a nuevos dispositivos o sistemas más modernos. Esto a la vez es una oportunidad de negocio mejorando el sistema para adelantar otros existentes en el mercado en un futuro.

Entre las oportunidades:

- Grandes empresas interesadas en el proyecto: si grandes cadenas de supermercados o empresas de transporte se interesan por este proyecto, la oportunidad de negocio es clara.

- Explotación comercial de los datos: las empresas integrarían entre sus datos estos nuevos parámetros, lo que permitiría aumentar la trazabilidad de sus mercancías

y por tanto de sus productos, integrándolos con el resto de datos de los que disponen actualmente.

3.5 Conclusiones

Una vez analizado el caso de uso y los problemas que se presentan, quedan establecidos los requisitos a cumplir en el diseño y que se implementarán en este trabajo. Es importante el análisis realizado puesto que con ello se ve de cerca el caso de negocio y los casos de uso, y por tanto los requisitos necesarios para que sea viable en el diseño e implementación. Con el caso de negocio que se ha analizado, queda claro que es un caso que reduce las pérdidas en las empresas por mercancías desechadas por encontrarse en mal estado, debido a unas malas condiciones de transporte, además de contribuir a un desarrollo sostenible. Además, al aumentar la trazabilidad, podrían verificar por ejemplo en un caso de reclamación por mal estado de un producto, si este ha sido causado en la cadena de transporte.

Por otro lado, tener estos datos permite a la empresa final integrar en su balance de cuentas las pérdidas reales por malas condiciones en el transporte y poder reclamarlo, si fuese el caso, a la empresa de transportes. Todo ello, conlleva una vez más a la conclusión de que, por un lado, reduce las pérdidas del cliente, y por otro, contribuye a la sostenibilidad al reducir el desperdicio de productos.

Una vez realizado el análisis, en el próximo capítulo se describe el diseño e implementación del sistema. Es importante haber realizado este análisis para tener en cuenta en el diseño los casos de uso a resolver y sus requisitos a cumplir.

4 Diseño e implementación

4.1 Introducción

En este capítulo se detalla el diseño realizado para este sistema, así como su implementación. El sistema implementado, se trata de una maqueta de lo que sería el sistema ideado, intentando que sea de bajo coste y simple con la finalidad de facilitar un posible posicionamiento en el mercado, pero cumpliendo todos los requisitos del análisis y el diseño realizado.

Primeramente, se presenta el esquema general de diseño del sistema y su descripción, pasando al detalle de cada uno de los elementos y su implementación, posteriormente se describe la programación necesaria, y por último el modelo de datos utilizado para el análisis.

4.2 Esquema general de la solución

En este trabajo, se ha optado por un diseño Sensores-Broker/Agregador-Servidor *cloud*, de tal forma que el dispositivo irá conectado al bróker en el SBC por wifi y transmitirá mediante protocolo MQTT para, posteriormente una vez agregados, enviarlos por LTE hacia un servidor *cloud* que tendrá una interfaz de consulta en un cuadro de mando para el usuario final.

La implementación, por tanto, será en su mayor parte inalámbrica, excepto la de los sensores en el SBC que va en la mercancía que irán soldados. El esquema de diseño para este trabajo se muestra en la siguiente imagen (Figura 4-1):

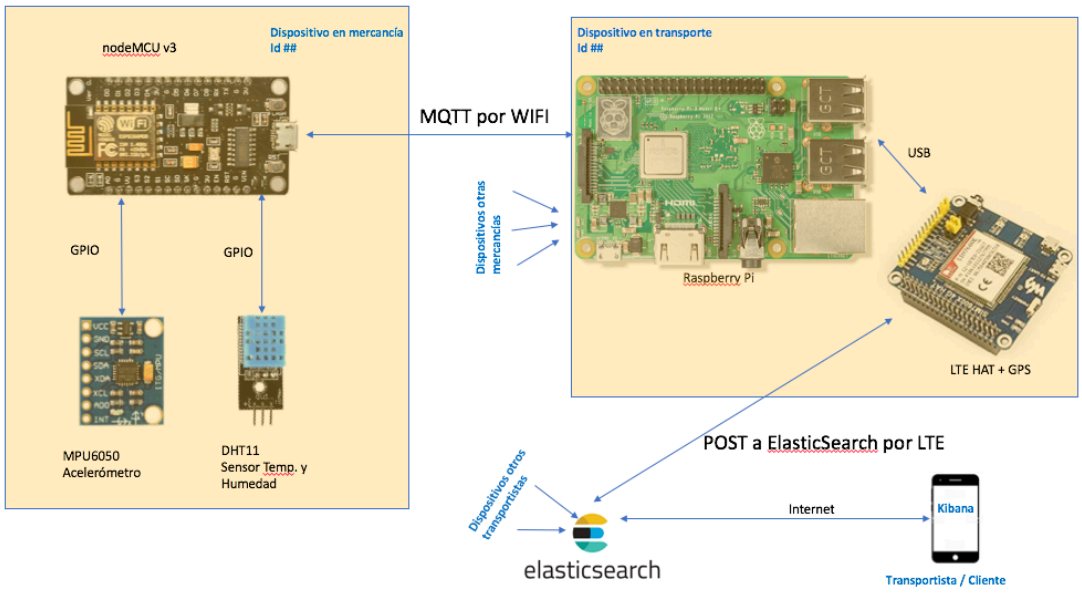

Figura 4-1: Esquema de diseño

Se puede observar que están por un lado, los sensores conectados al SBC instalado en la mercancía, que se encargará de recoger las medidas a través de los GPIOs y enviarlos por MQTT hacia la Raspberry Pi, el SBC principal instalado en el contenedor. Por otro lado, la Raspberry Pi, se encargará de enviar estos datos por LTE hacia Elasticsearch, para que puedan ser presentados en el cuadro de mando de una interfaz de usuario, en este caso Kibana, que será accesible desde Internet.

A nivel de físico y de enlace, en este trabajo en concreto se han elegido LTE y wifi. Las razones para elegir estas tecnologías y no el resto de las mencionadas en el capítulo Estado del arte, son por un lado que BLE o NB están pensados para casos de uso de baja energía con pocas transmisiones, y por tanto son idóneos para dispositivos de telemetría que transmiten medidas cada cierto tiempo y el resto permanecen en un "sueño profundo", ahorrando una cantidad de energía considerable y alargando la duración de la batería incluso durante años. Como este trabajo va a estar transmitiendo continuamente y en tiempo real, no serían protocolos en los que aprovechásemos sus ventajas y además encarecería el sistema por ser necesarios componentes compatibles. Por otro lado, Sigfox o LoRa, son protocolos menos extendidos, de los que hay muchos menos operadores que den servicio y que además son menos conocidos. De nuevo buscando la simplicidad del sistema con el objetivo de simplificar el mantenimiento e implementación del cliente final, y de reducir el coste al máximo, se decide descartar estos protocolos para este trabajo.

4.3 Elementos del diseño

4.3.1 Dispositivo en mercancía

El dispositivo en la mercancía se compone de los siguientes componentes: por un lado el sensor de temperatura y humedad y el acelerómetro para el movimiento, por otro lado el nodeMCU, un SBC encargado de recoger las medidas de los sensores a través de los GPIO (*General Purpose Input/Output,* entrada/salida de propósito general) y enviarlos por mensajes MQTT a la Raspberry vía WiFi.

Para medir temperatura y humedad, en este trabajo se ha optado por un sensor sencillo y de bajo coste, como es el modelo DHT11 (Figura 4-2). Cumple el objetivo de medida, es barato para la maqueta, y además tiene ambas funcionalidades en un solo sensor. Por otro lado, es un sensor utilizado ampliamente en maquetas con SBCs por su compatibilidad y fácil lectura de medidas. Este sensor se compone de un sensor de humedad capacitivo, un termistor y un circuito integrado que realiza la conversión de analógico a digital, enviando una señal digital con la lectura.

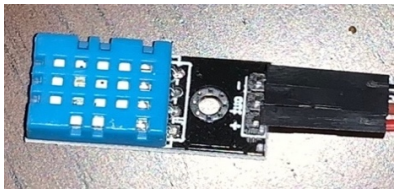

Figura 4-2: DHT11

Para monitorizar posibles golpes o caídas de la mercancía, se usará el acelerómetro MPU6050 (Figura 4-3). Con este acelerómetro lo que haremos será crear una alarma que indicará si en alguno de los ejes ha habido un movimiento superior un ángulo preestablecido, por ejemplo 45º para alertar si se vuelca la mercancía. Esta alarma indica que la mercancía ha sufrido una mala manipulación. De nuevo se opta por esta solución buscando sencillez y bajo coste, a la vez que un sensor amigable con un entorno con SBC para IoT. Está compuesto de giroscopio más acelerómetro.

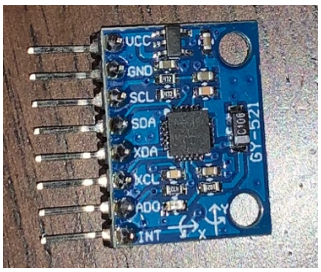

Figura 4-3: MPU6050

Ambos sensores irán integrados en un módulo nodeMCU (Figura 4-4) a través de los GPIOs, el cual se encargará de alimentar a los sensores, leer los datos de los mismos, y enviarlos vía MQTT hacia el bróker en la Raspberry Pi, conectados ambos vía wifi a través del AP (*Access Point,* punto de acceso) implementado en la propia Raspberry. El nodeMCU usado en este trabajo es el ESP8266, por su bajo coste y porque cumple los requisitos necesarios para la maqueta. Todo ello irá alimentado con una batería externa que permitirá el manejo de la mercancía sin interrumpir la comunicación siempre que se encuentre al alcance del SBC principal. La batería permite que el dispositivo permanezca activo durante 4 días, suficiente para este caso de uso en particular.

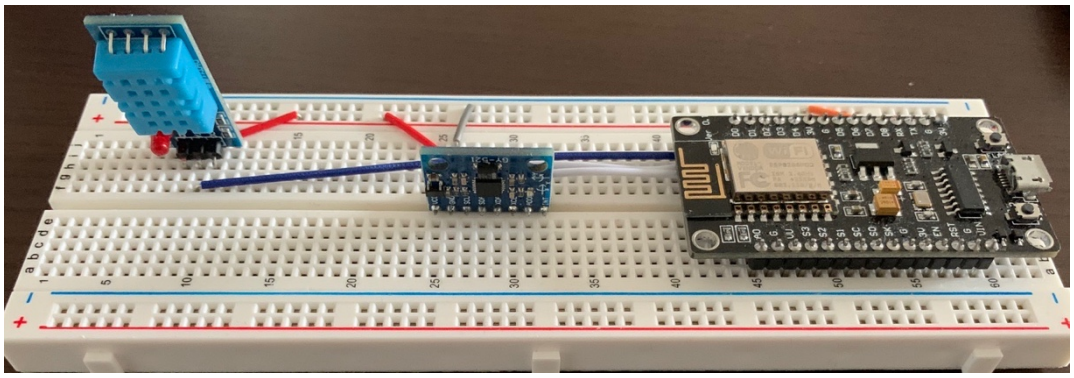

Figura 4-4: Protoboard con los sensores y el módulo nodeMCU

4.3.2 Dispositivo en contenedor

Como SBC principal, encargado de enviar los datos vía LTE a la nube y de recibir los datos de los sensores como bróker, se ha elegido para este proyecto una Raspberry Pi 3B+ (Figura 4-5).

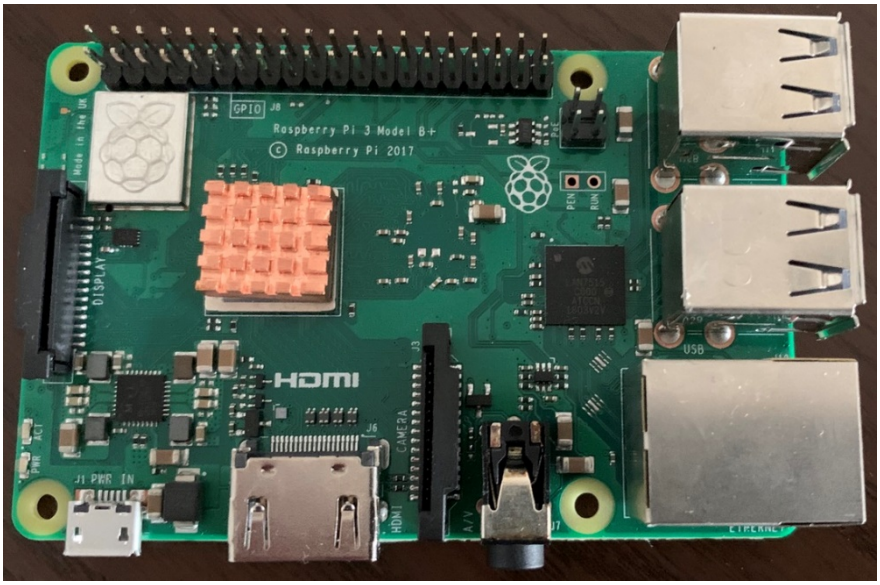

Figura 4-5: Raspberry PI 3B+

Se trata de un SBC de bajo coste, con alto rendimiento, y adecuado para casos de uso en IoT. Este SBC irá alimentado por la batería del contenedor, y contará con una conexión LTE a través de un LTE *HAT (Hardware Attached on Top,* hardware conectado en la parte superior) (Figura 4-6), un dispositivo que se conecta a través de los GPIO a la Rapsberry dotando a la misma de conectividad LTE y de señal GPS a la vez, lo que permite localizar el contenedor y a la vez transmitir los datos tanto de posición como de los sensores por LTE hacia Elasticsearch. Puesto que el *HAT* tiene doble funcionalidad, dotando de conectividad LTE a la Raspberry por un lado y facilitando la localización por otro, estará conectado a la Rasberry mediante USB para la conectividad LTE y mediante serial para la posición GPS.

Por otro lado, la Raspberry hace de AP wifi, creando así una red wifi a la que se conectará el nodeMCU automáticamente para transmitir los datos. Para crear esta red wifi como AP en la Raspberry se ha hecho uso de los programas DNSMasq y HostAPD siguiendo las instrucciones de la documentación de Raspberry [39].

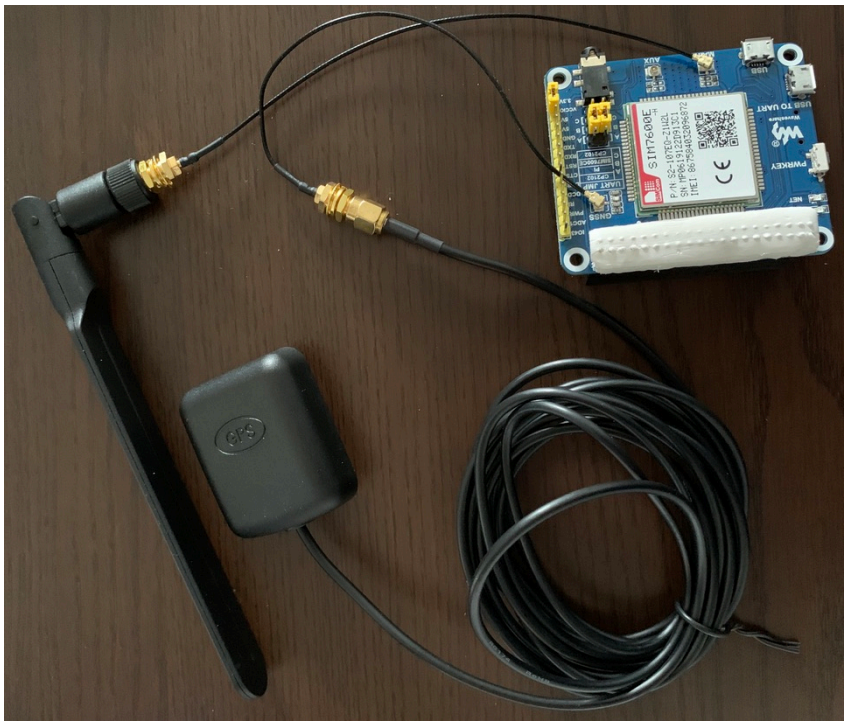

Figura 4-6: LTE HAT

La conexión elegida es LTE porque entre sus ventajas, destaca su rapidez, ancho de banda, menor latencia que en otras tecnologías de acceso móvil, y menor consumo de batería. Además, este módulo LTE cuenta con receptor GPS, lo que permite en un mismo dispositivo tener por un lado, conexión a Internet a través de la red móvil, y por otro, localización del dispositivo por GPS.

4.3.3 Comunicación entre contenedor y mercancía

Entre los protocolos a nivel de aplicación analizados en el capítulo de Estado del Arte, para la implementación de este trabajo se usa MQTT [8]. Las razones para ello, es que se trata de uno de los dos protocolos más usados en IoT (MQTT y CoAP [7]) y que por el diseño de la estructura de red que tiene este trabajo, se ajusta más a este protocolo, con múltiples sensores proporcionando información de forma continua, sin necesidad de que exista una solicitud para ello.

Si equiparamos el esquema de implementación de este trabajo a un esquema típico de MQTT [Figura 2-10], tendríamos como *Publisher* a las SBC de cada mercancía transmitiendo las medidas, y como Bróker MQTT y *subscriber* estaría el SBC central (Raspberry Pi), agregando datos y enviándolos hacia el servidor *cloud*. El bróker MQTT usado en este trabajo es Mosquitto [10], ya que se trata de un bróker de código abierto, y que está enfocado a su uso en SBC de baja potencia, como suele ser habitual en dispositivos IoT, por lo que es un bróker ligero y sencillo.

4.4 Programación del sistema

El nodeMCU encargado de leer las medidas de los sensores y transmitirlas al bróker vía MQTT, está programado en Arduino. Consta de un script que permite al SBC tomar la lectura de los sensores del puerto correspondiente, interpretarlas, y transmitirlas vía *publish* al bróker que estará escuchando en el topic correspondiente. Además, este script, se encarga de establecer la conexión del SBC con el bróker vía wifi. Mediante este script, se leen en bucle lecturas de los sensores, y las mismas son reportadas vía MQTT hacia la Raspberry Pi que está escuchando al otro lado. La frecuencia de lecturas será de 2,5 segundos para el DHT11 mientras que para el MPU6050 lo hará de forma constante, aunque sólo enviará el estado de alarma cada 10 segundos en el caso de detección de volcado, para evitar falsas alarmas y duplicados.

La Raspberry PI, SBC principal de la maqueta, consta de un script en Python que se encarga de, por un lado, suscribirse a los *topics*, y por otro, enviar los datos recibidos vía LTE hacia Elasticsearch. De esta forma, una vez en la nube, los datos pueden ser presentados en un cuadro de mando, en este caso con Kibana.

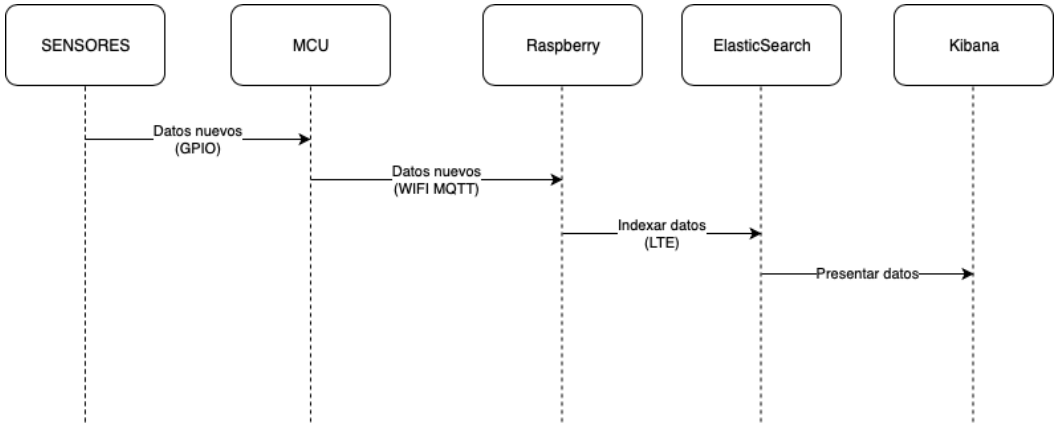

Figura 4-7: Diagrama de secuencia

En el diagrama de secuencia anterior (Figura 4-7), se puede observar el flujo desde que se recibe una nueva medida del sensor hasta que se representa en Kibana. Este flujo comienza desde los sensores (DHT11 y MPU6050), los cuales envían los datos de sus medidas a los GPIOs del nodeMCU. Una vez se reciben en el nodeMCU y son procesados, se envían por MQTT como mensaje *publish* vía wifi a la Raspberry Pi. La Raspberry se encargará de indexar estos datos junto a la posición GPS que recibe del LTE *HAT* en Elasticsearch por Internet vía LTE. Una vez en Elasticsearch, se presentarán en Kibana en un cuadro de mando.

La conectividad entre el nodeMCU y la Raspberry, en este caso por wifi, podría ser también por Bluetooth, puesto que la Raspberry lo permite. En este caso el módulo nodeMCU debería cambiarse por otro que lo permita, pero en cualquier caso el diagrama de

secuencia seguiría siendo el mismo. Cabe destacar que en este caso se supone una IP pública accesible en el servidor Elasticsearch, un PC para este trabajo de fin de máster. Pero bien es cierto que los servidores en muchas ocasiones, y más cuando se trata de proyectos M2M (*Machine to Machine*, máquina a máquina) o IoT, las IPs están detrás de un NAT (*Network Address Translation,* traducción de direcciones de red) o integradas en una VPN (*Virtual Private Network,* red privada virtual) por seguridad. En este caso, para que fuese accesible, cabrían varias posibilidades:

- Integrar la conectividad LTE en la VPN.
- Que tanto la conectividad del HAT como la del servidor de Elasticsearch sean móviles, y estén en una misma VPN móvil para M2M.
- Que el servidor sea una nube pública, como AWS (*Amazon Web Services,* servicios web de Amazon), Acens o Azure y conectar a través de IPSec.
- Abrir el puerto 9200 para la IP del servidor, el puerto en el que Elasticsearch estaría escuchando. Quizá esta no es la solución más segura pero sí la más económica y sencilla.

En cualquiera de estos casos, manejar IPs privadas no sería un problema puesto que seguiría habiendo visibilidad entre los equipos.

4.5 Modelo de datos para el almacenamiento de las medidas en Elasticsearch

Para este diseño, se utilizará Elasticsearch [13], que recibirá los datos vía LTE desde la Raspberry Pi, y posteriormente se usará Kibana, para presentar en el cuadro de mando los datos recibidos y que sean accesibles desde cualquier lugar por Internet. En Elasticsearch se almacenan las medidas de los sensores diferenciados por *topic*, de tal forma que luego los datos se pueden tratar filtrando según el sensor que nos interesa representar.

Estos datos se representan en gráficas para el caso de temperatura y humedad, en forma de contador para las alarmas de golpes, y en un plano en el caso de la localización. De esta forma, son visibles y manejables desde el cuadro de mando sin necesidad de estudio previo. El modelo de datos utilizado es el siguiente (Figura 4-8):

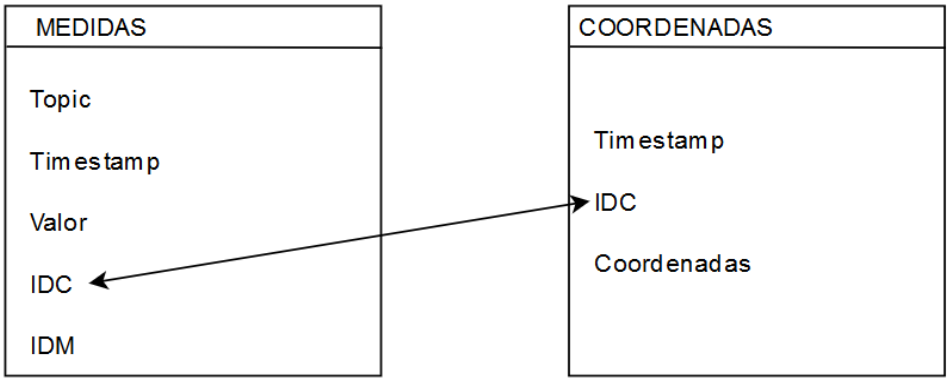

Figura 4-8: Modelo de datos

De tal forma tenemos dos índices, por un lado, el de "MEDIDAS" que incluirá los datos de alarmas, humedad y temperatura, y por otro el de "COORDENADAS". Ambos índices incluyen un campo llamado IDC (identificador del contenedor), que permitirá filtrar los datos por un contenedor concreto. Además, el índice de "MEDIDAS" incluye un campo llamado "IDM" (identificador de mercancía), que permite filtrar los datos de los sensores por una mercancía concreta entre las que porta el contenedor. Con este modelo de datos, se puede de una forma visual y rápida comprobar el estado de una mercancía, su posición, y en caso de detectar una anomalía, como por ejemplo un exceso de humedad, podemos ver si se trata únicamente de esa mercancía, o por el contrario afecta a todo el contenedor, ya que podemos filtrar por mercancía, y posteriormente por contenedor para verificar si los datos coinciden o discrepan.

4.6 Conclusiones

En este trabajo, se ha buscado una maqueta de bajo coste y sencilla que cubra los objetivos iniciales para el caso de uso elegido, es decir, monitorizar una mercancía durante su transporte. Es por ello que los componentes elegidos, se ha intentado que sean de fácil implementación, para facilitar el mantenimiento de los mismos, y de bajo coste, ya que no se requiere de altas prestaciones para este caso de uso.

Por otro lado, las conectividades elegidas, se trata de conectividades accesibles, y sencillas de implementar, aunque como se explica, se podrían valorar otras, para resolver conflictos de direcciones IP y posibles amenazas de Internet. Con el diseño implementado en esta maqueta, se resuelve el problema y los casos de uso planteados, pero podría implementarse con otras variantes y seguiría siendo igualmente válido. Llevado a un caso de negocio final como el que se plantea en análisis, lo más recomendable sería integrarlo todo en una VPN por seguridad o que estuviese en una *cloud* pública conectado por IPSec. Hay que tener en cuenta que los datos de localización son sensibles, ya que en el caso de los robos saber exactamente la localización del contenedor con las mercancías es potencialmente peligroso.

Se trata de un sistema que abarca múltiples y diferentes tecnologías, así como lenguajes de programación, lo que implica trabajo de documentación y análisis previo, así como un buen diseño e implementación, para conseguir el enlace entre los diferentes componentes del sistema y que sea posible la comunicación entre todos ellos. Esto posibilita que la fase descrita en el siguiente capítulo de validación y pruebas pueda llevarse a cabo.

5 Validación y pruebas

5.1 Introducción

Una vez realizado el diseño completo e implementación de la maqueta, se han realizado pruebas de funcionamiento, conectividad y consumo energético para verificar que los requisitos y, por tanto, objetivos del trabajo, se han alcanzado y cumplido.

Es importante la realización de pruebas puesto que, aunque se trata de una maqueta, la misma se podría presentar a un posible cliente interesado, y cualquier fallo puede significar la diferencia entre el éxito o el fracaso del caso de negocio. Por ello, es necesario probar que todo lo diseñado e implementado, funciona correctamente, pieza por pieza.

En este capítulo se describen las diferentes fases de validación y pruebas del sistema, comenzando por la validación funcional por componentes, posteriormente realizando pruebas de conetividad, rendimiento y consumo, y por último las conclusiones extraídas de las pruebas realizadas.

5.2 Validación funcional

Se prueban las diferentes partes y fases de la maqueta, dividiéndolo por:

- nodeMCU y sensores.
- Raspberry Pi y LTE *HAT*.
- Elasticsearch y Kibana.

5.2.1 nodeMCU y sensores

En esta etapa se comprueba que los sensores miden correctamente, contrastando los valores para validarlo. Se comprueban datos de temperatura, humedad y ángulos de movimiento. También se comprueba que el margen de error de estas medidas no influye en la finalidad del trabajo, es decir, este margen de error no da “falsas alarmas” en el transporte de la mercancía. En esta etapa también se comprueba que el módulo nodeMCU recibe dichos datos, que consigue conectarse por wifi y transmite los datos recibidos de los sensores vía wifi por MQTT a la Raspberry (Figura 5-1) según la programación en Arduino del dispositivo.

```
pi@raspberrypi:~ $ sudo tshark -i wlan0
Running as user "root" and group "root". This could be dangerous.
Capturing on 'wlan0'
    1 0.000000000  10.3.141.86 → 10.3.141.1   MQTT 77 Publish Message [12345temperatura]
    2 0.000140262   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=24 Win=29200 Len=0
    3 0.022981464  10.3.141.86 → 10.3.141.1   MQTT 74 Publish Message [12345humedad]
    4 0.023084486   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=44 Win=29200 Len=0
    5 2.456757055  10.3.141.86 → 10.3.141.1   MQTT 79 Publish Message [12345temperatura]
    6 2.456888150   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=69 Win=29200 Len=0
    7 2.468332948  10.3.141.86 → 10.3.141.1   MQTT 76 Publish Message [12345humedad]
    8 2.468400762   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=91 Win=29200 Len=0
    9 5.002184916  10.3.141.86 → 10.3.141.1   MQTT 77 Publish Message [12345temperatura]
   10 5.002309240   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=114 Win=29200 Len=0
   11 5.007541844  10.3.141.86 → 10.3.141.1   MQTT 74 Publish Message [12345humedad]
   12 5.007612730   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=134 Win=29200 Len=0
   13 6.829813228  10.3.141.86 → 10.3.141.1   MQTT 70 Publish Message [12345Alarma]
   14 6.829946979   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=150 Win=29200 Len=0
   15 7.460740008  10.3.141.86 → 10.3.141.1   MQTT 79 Publish Message [12345temperatura]
   16 7.460812456   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=175 Win=29200 Len=0
   17 7.465539380  10.3.141.86 → 10.3.141.1   MQTT 76 Publish Message [12345humedad]
   18 7.465607610   10.3.141.1 → 10.3.141.86  TCP 54 1883 → 64536 [ACK] Seq=1 Ack=197 Win=29200 Len=0
^C18 packets captured
```

Figura 5-1: Tráfico MQTT

5.2.2 Raspberry pi y LTE HAT

En esta etapa, se comprueba que la Raspberry Pi funciona correctamente una vez conectada a la fuente de alimentación. Posteriormente, se comprueba con el comando *ifconfig* que la interfaz wifi del AP (Figura 5-2) que dará acceso al nodeMCU está levantada, funcionando, y que no ha variado su IP local. En este punto se comprueba a la vez la interfaz LTE (Figura 5-3) con este mismo comando, comprobando de igual manera que está levantada y permite la salida a Internet. Esta conectividad será la que permita trasladar los datos a Elasticsearch y, por tanto, a Kibana y al cuadro de mando.

```
pi@raspberrypi:~/Desktop $ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.3.141.1  netmask 255.255.255.0  broadcast 10.3.141.255
        inet6 fe80::50e:35a2:718:cdb4  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:c4:fa:b7  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 255 (255.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 42  bytes 5664 (5.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 5-2: Interfaz wifi AP Raspberry Pi

```
pi@raspberrypi:~/Desktop $ ifconfig wwan0
wwan0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.13.198.191  netmask 255.255.255.128  destination 10.13.198.191
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 4  bytes 764 (764.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35  bytes 5923 (5.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 5-3: Interfaz LTE Raspberry Pi

Además, se prueban los scripts Python que corren en la Raspberry enviando los datos a Elasticsearch; por un lado, los datos del nodeMCU y por otro, las posiciones GPS que el propio LTE HAT facilita. Para obtener estas posiciones GPS, se envían comandos AT al LTE HAT y devuelve la posición en forma de coordenadas como se observa en la siguiente figura (Figura 5-4).

```
AT+CGPSINFO
+CGPSINFO: 4020.373377,N,00340.550213,W,020220,182809.0,559.3,0.0,337.3
```

Figura 5-4: Comando AT GPS

Usando Wireshark, se comprueba que las trazas HTTP hacia Elasticsearch (Figura 5-5) no devuelven error y que estos datos quedan indexados en Elasticsearch, lo que prueba que los scripts funcionan correctamente.

Time	Source	Destination	Protocol	Length	Info
0.000000	176.83.10.198	192.168.1.40	HTTP	381	POST /dat/_doc HTTP/1.1 (application/json)
0.405164	192.168.1.40	176.83.10.198	HTTP	373	HTTP/1.1 201 Created (application/json)
0.480006	176.83.10.198	192.168.1.40	HTTP	377	POST /dat/_doc HTTP/1.1 (application/json)
1.247608	192.168.1.40	176.83.10.198	HTTP	373	HTTP/1.1 201 Created (application/json)

Figura 5-5: Indexando datos en Elasticsearch

5.2.3 Elasticsearch y Kibana

Finalmente, se comprueba que Elasticsearch está recibiendo en los índices creados los datos correspondientes, y que pueden ser presentados en Kibana. Desde Kibana se comprueba la estructura del *mapping* del índice, que se están recibiendo datos en el formato correcto, y se vuelcan al Dashboard (Figura 5-6) creado correctamente.

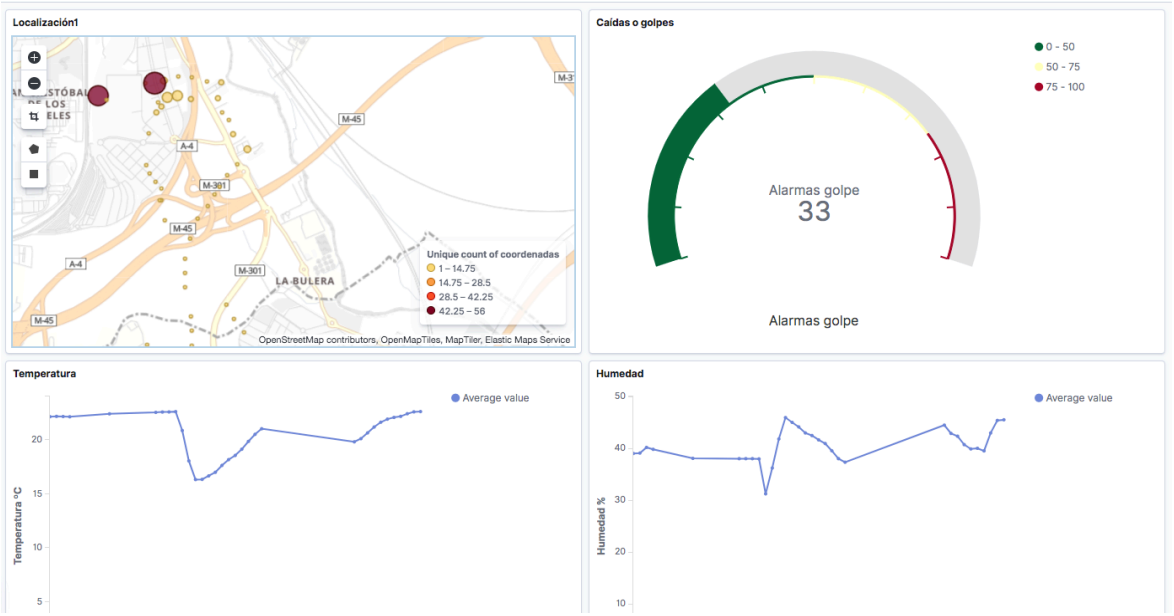

Figura 5-6: Dashboard Kibana

Además, se comprueba que el modelo de índice es el *mapping* realizado, y que los datos llegan en el formato correcto. Esto permite después filtrar por IDC e IDM (Figura 5-7):

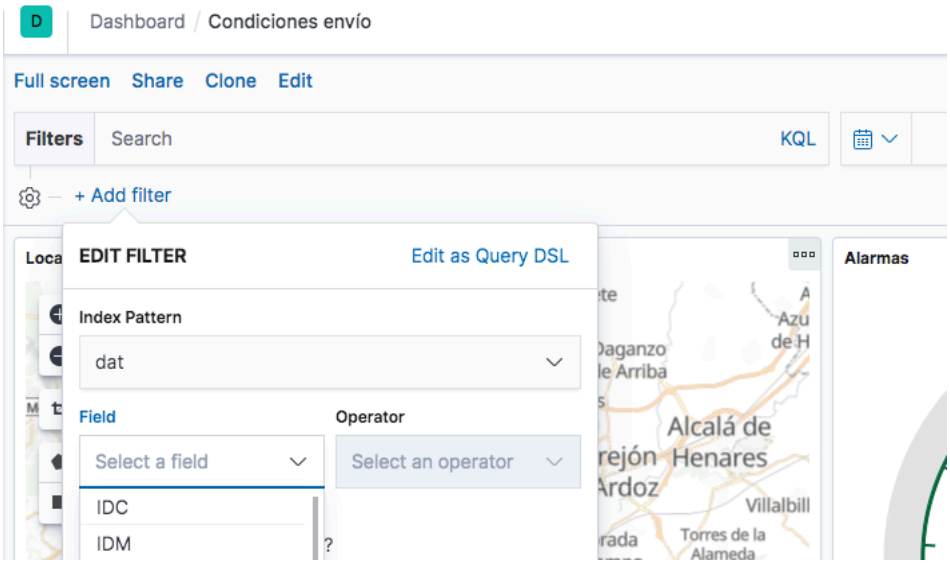

Figura 5-7: Filtros Kibana

5.3 Pruebas de conectividad

En la prueba funcional ya es posible deducir que la conectividad es correcta, pero en pasos anteriores se comprueba paso a paso y capturando tráfico con Wireshark, las siguientes conectividades:

- Wifi: se comprueba que la conexión vía wifi que permite establecer tráfico MQTT es correcta.

- MQTT: se comprueba que existe conexión y tráfico entre el módulo nodeMCU y la Raspberry y que no hay error en la comunicación.

- LTE: una vez configurado el módulo, se comprueba que hay conexión a Internet desde la Raspberry desactivando las interfaces ethernet y wifi. Además, se comprueba que desde la Raspberry es posible alcanzar la IP pública del servidor de Elasticsearch a través de Internet.

5.4 Pruebas de rendimiento

Se comprueba que las actualizaciones de posición GPS son suficientemente frecuentes para poder hacer seguimiento del recorrido de la mercancía, ya que el HAT de la Raspberry da la localización cada 2 segundos. Además, se comprueba que el margen de error no afecta a la trazabilidad del contenedor, ya que tiene un margen de error entorno a 200 metros. Aunque este sistema de localización GPS (integrado en el HAT de la Raspberry) no es el más preciso, ahorra un coste importante respecto a lo que tendría un dispositivo aparte para ello.

Por otro lado, se prueba el *throughput* necesario para un contenedor. Calculado para las 25 mercancías que suele llevar un contenedor de transporte, y teniendo en cuenta que se envía un paquete de entorno a 380 bytes por cada dato nuevo indexado, bien sea temperatura, humedad, alarma o localización, al cabo de un mes transmitimos entorno a 12,5GB. Por tanto, es necesaria una tarjeta SIM (*Susbcriber Identity Module*, módulo de identificación de abonado) para M2M (SIM con rango de numeración especial de 13 dígitos establecidos por la Comisión Nacional de los Mercados y la Competencia [37]) con un plan de datos superior a esta cifra.

Por otro lado, teniendo en cuenta las frecuencias de indexado de datos, para una flota de 1.000 contenedores como ejemplo, estaríamos indexando 13.000 datos por segundo, lo que significa que habría que implementar un *cluster* de nodos de Elasticsearch para poder abordarlo, ya que un único nodo es capaz de indexar entorno a 3.000 datos por segundo.

5.5 Pruebas de consumo energético

Se realiza prueba en tiempo real con una batería con capacidad de 11400 mAh conectada al SBC de la maqueta que iría en la mercancía, y permanece activo enviando datos durante 4 días, lo que implica un consumo de 118,75 mA. Esta duración podría incrementarse con baterías externas de más capacidad, aunque en principio en un caso de cliente como el de una cadena de supermercados no suelen ser trayectos tan largos los que realiza la mercancía por tierra.

Para la Raspberry no es necesario realizar este tipo de pruebas ya que irá conectado a la batería del contenedor/vehículo, por lo que no habrá problemas de alimentación.

5.6 Conclusiones

Con las pruebas realizadas, queda validado el funcionamiento de la maqueta de este trabajo. Se ha verificado que todo funciona según lo previsto en el diseño, y que la implementación está bien realizada. Observando el *dashboard* durante las pruebas, se revisan objetivos y requisitos comprobando que la finalidad del proyecto se cumple. Así, posicionándose como cliente final, de un rápido vistazo es posible ver los datos globales de la cadena de transporte. Los filtros son sencillos, simplemente introduciendo un identificador, y los datos quedan filtrados de forma rápida.

Durante la fase de pruebas, se han detectado pequeños fallos que han ido corrigiéndose, como por ejemplo errores en el formato en que se envían los datos a Elasticsearch, fallos en la conectividad por mala configuración, o errores de medida en alarmas por golpe. Sin estas pruebas, al mostrarlo a un posible cliente interesado, hubiesen llevado este sistema al fracaso. Por ello, es realmente importante validar que el diseño e implementación funcionan según lo esperado.

Todo ello nos lleva a las conclusiones finales, evaluando así el sistema realizado en este trabajo, y líneas de trabajo futuro que puedan mejorarlo.

6 Conclusiones y trabajo futuro

6.1 Resumen

En este trabajo se ha implementado una maqueta que monitoriza el estado en el que se transportan mercancías. De esta forma, podremos saber en cualquier momento y desde cualquier dispositivo conectado a Internet, como un *smartphone* o un ordenador, la localización del envío y las condiciones de temperatura, humedad y estabilidad en las que se está transportando. Así, si una mercancía llega a su destino dañada por la cadena de transporte, se podrá saber que ha sido así debido a un golpe, un exceso de temperatura, o de humedad y reclamarlo al transportista.

6.2 Conclusiones

Para este trabajo han sido de gran utilidad asignaturas del máster como Planificación de Redes, Sensores y Actuadores, Gestión de Redes, Tecnologías y Servicios de Internet o Proyectos en Ingeniería de Telecomunicación.

De este trabajo de fin de máster se extraen varias conclusiones. Primeramente, que existe un problema real que con IoT podemos resolver parcialmente, y no por ello despreciable. Esto es, el desecho de productos por un mal transporte. Todos hemos sufrido un paquete que llega golpeado, una fruta en el supermercado que al abrirla está en mal estado, o una bandeja de carne que, por una mala cadena de frío, cuando la abrimos no es apta para su consumo. Con este sistema no evitamos que esto ocurra en primera instancia, pero si ocurre, sabremos si ha sido culpa del transportista, y teniendo ese dato, el transportista será responsable de ello, asumirá las consecuencias y corregirá ese error para las siguientes veces, lo que desencadena en que este error no vuelve a ocurrir y por tanto no sigamos desechando productos.

Por otro lado, es evidente que el IoT es una línea muy interesante para desarrollar nuevos dispositivos ya que, aunque ha avanzado mucho, es prácticamente aplicable a cualquier cosa, y dotar de conectividad a algo es interesante y útil para las personas, ya que siempre facilita alguna tarea que de otra forma harían los humanos.

Por último, encontramos que el mundo IoT está fuertemente ligado al mundo *Big Data*, ya que cualquiera que sea la utilidad de un dispositivo IoT, va a ser capaz de generar nuevos datos. Datos que para ser de utilidad tienen que ser almacenados, analizados y presentados correctamente.

6.3 Trabajo futuro

Para este trabajo, se proponen las siguientes vías de trabajo futuro para mejorar el sistema. Primeramente, en cuanto a seguridad, se plantea que, en caso de robo, los datos de la wifi que da acceso a Elasticsearch desde las diferentes Raspberry serían vulnerables. Por tanto, habría que generar un mecanismo de cambio de clave de forma masiva. En cuanto a sensorización se refiere, se pueden valorar otros tipos de sensores de interés según el tipo de mercancía del cliente final, como podrían ser de luz o de presión, por ejemplo, midiendo la presión ejercida en el trayecto sobre un material frágil como podría ser un cristal, o midiendo el peso de la mercancía enviada, de tal forma que si hay una variación, se puede tratar de una pérdida en el trayecto o robo.

También se puede valorar la posibilidad de incorporar otros modelos de sensores más modernos con mejores características, otras conectividades como alguna de las que se han mencionado en esta memoria, u otras formas de análisis *Big Data* y de presentación en *Dashboard*.

Otra línea de trabajo futuro para este sistema sería llevarlo a producción, y sacarlo al mercado. Se trata de un sistema innovador, y potencialmente interesante para clientes finales como cadenas de supermercados o transportistas.

Referencias

[1] Real Academia Española, "Internet de las cosas", https://dej.rae.es/lema/Internet-de-las-cosas [Fecha de acceso 01/02/2020]

[2] Scully, Padraig, "The Top 10 IoT Segments in 2018 – based on 1,600 real IoT projects", https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/ [Fecha de acceso: 13/12/2019]

[3] Universidad Autónoma de Madrid, "La UAM lanza el primer bus autónomo universitario de España", https://www.uam.es/UAM/BUS-AUTONOMO-UAM-2020/1446795199280.htm [Fecha de acceso 01/01/2020]

[4] Samsara, "Dash cam", https://www.samsara.com/es/fleet/dash-cam [Fecha de acceso: 13/12/2019]

[5] Smarter, "Fridge cam", https://www.smarter.am/fridgecam [Fecha de acceso: 13/12/2019]

[6] Standardization, I, "ISO/IEC 7498-1: 1994 information technology–open systems interconnection–basic reference model: The basic model.", International Standard ISOIEC 74981, 1996.

[7] Zach Shelby, Klaus Hartke, and Carsten Bormann, "The constrained application protocol (CoAP)", IETF Request for Comments 7252, 2014. https://tools.ietf.org/html/rfc7252.

[8] Andrew Banks, Ed Briggs, Ken Borgendale, Rahul Gupta (eds.), "MQTT version 5.0", MQTT Standard. Oasis. 07 March2019. http://docs.oasis-open.org/mqtt/mqtt.

[9] Agencia Estatal Boletín Oficial del Estado, "Resolución de 7 de octubre de 2019, de la Dirección General de Trabajo, por la que se registra y publica el XIX Convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos.", https://www.boe.es/eli/es/res/2019/10/07/(8), Octubre 2019 [Fecha de acceso 02/02/2020]

[10] Eclipse Fundation, "ECLIPSE MOSQUITTO, an open source MQTT broker", https://mosquitto.org [Fecha de acceso 01/02/2020]

[11] Elasticsearch B.V., "Elastic Stack", https://www.elastic.co/es/what-is/elk-stack [Fecha de acceso 01/02/2020]

[12] Rute C.Sofia, "An overview on the Evolution of Communication Approaches", COPELABS, University Lusófona, 2018.

[13] Elasticsearch, B.V., "Open Source Search & Analytics · Elasticsearch | Elastic". https://www.elastic.co/es/ [Fecha de acceso: 07/03/2019]

[14] Vasileios Karaggianis, Periklis Chatzimisios, Francisco Vazquez-Gallego, Jesus Alonso-Zarate, "A survey on application layer protocols for the Internet of Things", Transaction on IoT and Cloud computing, vol. 3, no. 1, 2015, pp. 11-17.

[15] Erwin Sacoto-Cabrera, Pablo Gallegos-Segovia, Gabriel Leon-Paredes, Jorge Rodriguez-Bustamante, Gabriela Arevalo-Quishpi, "Internet of Things: Informatic system for metering with communications MQTT over GPRS for smart meters", CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication, Chile, 2017, pp. 1-6,DOI: 10.1109/CHILECON.2017.8229598.

[16] Diego Ortiz, Diego S. Benítez, Walter Fuertes, Jenny Torres, "On the use of low cost sensors for the implementation of a real-time air pollution monitoring system using wireless sensor networks", Proc. 2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2018). Ixtapa, Mexico, 7-9 November 2018. DOI: 10.1109/ROPEC.2018.8661479.

[17] Robert Bosch GmbH, "the Bosch IoT Shuttle", https://www.bosch-mobility-solutions.com/en/highlights/connected-mobility/connected-vehicle/ [Fecha de acceso: 13/12/2019]

[18] Joshi, Ramesh C., Gupta Brij B., "Security, Privacy, and Forensics Issues in Big Data", IGI Global, 2019.

[19] Chanjoo Lee, Suyun Park, Yoonha Jung, Youngji Lee, Mariah J. Mathews, "Internet of Things: Technology to Enable the Elderly", Second IEEE International Conference on Robotic Computing (IRC), 2018, DOI: 10.1109/IRC.2018.00075.

[20] Arkady Zalavsky, Prem Prakash Jayaraman, "Discovery in the Internet of Things. Ubiquity", 2015, pp. 1–10, DOI: 10.1145/2822529.

[21] Xiaomin Li, Di Li, Jiafu Wan, Athanasios V. Vasilakos, Ching-Feng Lai, "A review of industrial wireless networks in the context of Industry 4.0", Wireless Netw 23, pp. 23–41, 2017, DOI : 10.1007/s11276-015-1133-7.

[22] Roberto Leal, Leonel Santos, Leandro Vieira, Ramiro Gonçalves, Carlos Rabadão, "MQTT Flow Signatures for the Internet of Things", 14th Iberian Conference on Information Systems and Technologies (CISTI), 2019, DOI: 10.23919/cisti.2019.8760849.

[23] P.P. Ray, "A Survey on Internet of Things Architectures", EAI Endorsed Transactions on Internet of Things, 2010, DOI: 10.1016/j.comnet.2010.05.010.

[24] Luigi Atzori, Antonio Iera, Giacomo Morabito, "The Internet of Things: A Survey", Computer Networks, 2010, pp. 2787-2805, DOI: 10.1016/j.comnet.2010.05.010.

[25] Europa Press, “El 49% de los productos aptos para el consumo va a la basura y el 24% se dona en España, según un estudio”, Madrid, 2018, https://www.europapress.es/epsocial/responsables/noticia-49-productos-aptos-consumo-va-basura-24-dona-espana-estudio-20180313180943.html [Fecha de acceso: 08/11/2019]

[26] Organización de las Naciones Unidas, “Objetivo 12: Garantizar modalidades de consumo y producción sostenibles”, https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/ [Fecha de acceso: 10/01/2020]

[27] Bluetooth SIG, Inc, “Bluetooth”, https://www.bluetooth.com/es/ [Fecha de acceso: 10/01/2020]

[28] Sigfox España, “Sigfox”, https://www.sigfox.es, [Fecha de acceso: 10/01/2020]

[29] LoRa Alliance, “LoRa”, https://lora-alliance.org , [Fecha de acceso: 10/01/2020]

[30] B. P. Crow, I. Widjaja, J. G. Kim, P. T. Sakai, "IEEE 802.11 Wireless Local Area Networks”, IEEE Communications Magazine, vol. 35, no. 9, pp. 116-126, 1997, DOI: 10.1109/35.620533

[31] 3GPP, 3rd Generation Patnership Project, “LTE”, https://www.3gpp.org/technologies/keywords-acronyms/98-lte, [Fecha de acceso: 10/01/2020]

[32] 3GPP, 3rd Generation Patnership Project, “Standardization of NB-IOT completed”, https://www.3gpp.org/news-events/3gpp-news/1785-nb_iot_complete, [Fecha de acceso: 10/01/2020]

[33] Eclipse Fondation Inc, “IoT Developer Survey Results”, pp. 39, 2018, https://iot.eclipse.org/resources/iot-developer-survey/iot-developer-survey-2018.pdf

[34] The Apache Software Foundation, “Apache Hadoop”, https://hadoop.apache.org , [Fecha de acceso: 14/10/2019]

[35] MongoDB Inc, “MongoDB”, https://www.mongodb.com/es, [Fecha de acceso: 14/10/2019]

[36] Gary Davis, "2020: Life with 50 billion connected devices", IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2018, pp. 1-1, DOI: 10.1109/ICCE.2018.8326056

[37] Comisión Nacional de los Mercados y la Competencia, “Acuerdo por el que se emite informe sobre la propuesta de resolución de la secretaría de estado para la sociedad de la información y la agenda digital por la que se modifica la resolución de 27 de mayo de 2013, relativa a los rangos de numeración atribuidos a los servicios de comunicaciones móviles”, Madrid, 2018, https://www.cnmc.es/sites/default/files/1977959_3.pdf

[38] M.U. Farooq, Muhamed Waseem, Sadia Mazhar, Anjum Khairi, Talha Kamal, “A Review on Internet of Things (IoT)”, International Journal of Computer Applications, 2015, https://www.researchgate.net/profile/Muhammad_Farooq75/publication/273693976_A_Review_on_Internet_of_Things_IoT/links/5508ac290cf26ff55f83af53.pdf

[39] Raspberry Pi Foundation, “Setting up a Raspberry Pi as a Wireless Access Point”, https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

Anexos

A Presupuesto

Tabla A-1: Coste componentes

Costes componentes			
Componente	**Observaciones**	**Coste maqueta**	**Coste Wholesale**
Raspberry Pi		60 €	40 €
Protoboard y accesorios		14 €	
DHT11	*Sensor temperatura y humedad*	2 €	0,80 €
MPU6050	*Acelerómetro*	1 €	0,45 €
Power Bank	*11400 mAh*	28 €	6 €
MCU		8 €	8 €
Módulo LTE+GPS		78 €	50 €
Rugerizado			3 €
SIM LTE			10€/mes

Estos costes engloban, por un lado el coste que ha tenido la maqueta de este proyecto, y por otro, el coste que tendría en un caso real el sistema comprando los componentes como mayorista.

Tabla A-2: Coste desarrollo

Costes desarrollo				
Desarrollo	**Observaciones**	**Horas**	**€/h**	**Coste**
Script MPU	*MQTT*	50	20 €	1.000 €
Script Raspberry	*ELK*	30	20 €	600 €
Costes I+D		300	20 €	6.000

Los costes de desarrollo engloban costes de Investigación y Desarrollo, tenidos en cuenta como las 300 horas de este trabajo de fin de máster.

Tabla A-3: Coste mantenimiento

Costes mantenimiento					
Desarrollo	**Observaciones**	**Horas**	**€/año**	**Coste**	**MTTF**
Reemplazo sensores DHT11	*Por fallo, se reemplaza dispositivo y repara posteriormente técnico permanente*		0,8	0,80 €	1 año
Reemplazo sensores MPU6050			0,45	0,45 €	1 año
Reemplazo tarjeta de memoria			12	6,00 €	6 meses
Reemplazo Raspberry			12	60,00 €	5 años
Reemplazo MCU			2,67	8,00 €	3 años
Reemplazo baterías	*Ciclos de carga / fallo*		9,33	28,00 €	5 años (500 ciclos)
Formación reemplazo baterías	*1h/mes*	12	180,00		
Actualizaciones	*Seguridad/SO/supervisión*	Permanente (8x5)	29.000		*Anuales*
		Coste anual	53.409,25 €		

En el coste de mantenimiento anual se tienen en cuenta el coste de reemplazo de los diferentes componentes, así como de formación de personal para reemplazo de baterías y de actualizaciones y mantenimiento del sistema. Las baterías serían reemplazadas por los propios transportistas tras un curso básico para evitar una mala manipulación.

Tabla A-4: Caso de negocio supermercado

CASO SUPERMERCADO		
	Coste mayorista	
Palés/camión	25	
Camiones en flota	1000	
Total uds senores	25000	
Total uds Rasp Pi	1000	
Coste	553.850,00 €	Sin SIM ni Cloud
Coste inicial de implemetación		

En el caso por ejemplo, de una cadena de supermercados, con una flota de 1.000 contenedores para transporte de mercancías en la cadena de suministro, el precio inicial de implementación de este sistema sería de 553.850€.

Tabla A-5: Coste de equipamiento de un contenedor

Coste Unidad Completa		
Maqueta	193 €	
Wholesale	548 €	*Equipar un camión*

El coste de una unidad completa del sistema, es decir, un contenedor o camión con sus 25 mercancías equipadas, tendría un coste de 548€. Por otro lado, el coste total de la maqueta ha sido de 193€.

B Código

En este anexo se detalla el código programado en cada uno de los componentes.

Para enviar los datos recibidos por MQTT hacia Elasticsearch se utiliza el siguiente código en Python:

```
mqttServer="10.3.141.1"
mqttPort="1883"
IP_MAC="80.29.157.202"
PORT_MAC="9200"
channelSubs="#"

import paho.mqtt.client as mqtt
from datetime import datetime
from elasticsearch import Elasticsearch

IDC=str("12A34B")

def on_connect(client, userdata, flags, rc):

    print("Connected with result code "+str(rc))

    client.subscribe(channelSubs)


def on_message(client, userdata, msg):

    print(msg.topic+" "+str(msg.payload))



    dato=float(msg.payload)

    es.index(index="dat" ,body = {"topic": msg.topic[5:len(msg.topic)],
"timestamp": datetime.utcnow(), "value": dato, "IDC": IDC, "IDM":
msg.topic[0:5]})



es = Elasticsearch([{'host': IP_MAC, 'port': PORT_MAC}],
verify_certs=True)
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(mqttServer,mqttPort, 60)
client.loop_forever()

# creación de mapping

"""    mapping = {

        "mappings":{

            "properties":{

                "timestamp": {"type":"date"},
```

```
                "topic": {"type": "text"},
                "value": {"type": "float"},
                "IDC": {"type": "text"},
                "IDM": {"type": "text"}
        }
        }
        }

    es.indices.create(index='dat', body=mapping)
"""
```

Para recibir la posición GPS mediante comandos AT y convertir las coordenadas al formato que usa Elasticsearch, para posteriormente enviarlas, se usa el siguiente código en Python:

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import RPi.GPIO as GPIO

import serial
import time
IP_MAC="80.29.157.202"
PORT_MAC="9200"
from datetime import datetime
from elasticsearch import Elasticsearch
IDC=str("12A34B")
ser = serial.Serial('/dev/ttyS0',115200)
ser.flushInput()

power_key = 6
rec_buff = ''
rec_buff2 = ''
time_count = 0

def send_at(command,back,timeout):
    rec_buff = ''
    ser.write((command+'\r\n').encode())
    time.sleep(timeout)
    if ser.inWaiting():
        time.sleep(0.01 )
        rec_buff = ser.read(ser.inWaiting())
    if rec_buff != '':
        if back not in rec_buff.decode():
            print(command + ' ERROR')
            print(command + ' back:\t' + rec_buff.decode())
            return 0
        else:
            print(rec_buff.decode())
            a=rec_buff.decode()
            print(a)
```

```
            if len(a)>20:
                if a[27].isnumeric():
                    ax=a[25:36]
                    ay=a[40:51]
                    print (ax)
                    print (ay)
                    print (a[37])
                    print (a[52])
                    if a[37] == 'S':
                        lat=(-
1)*(float(ax[0:2])+(float(ax[2:4])/60)+(float(ax[5:7]+'.'+ax[7:10])/3600)
)
                        print (lat)
                    else:

lat=(float(ax[0:2])+(float(ax[2:4])/60)+(float(ax[5:7]+'.'+ax[7:10])/3600
))
                        print (lat)
                    if a[52]=='W':
                        long=(-
1)*(float(ay[0:2])+(float(ay[2:4])/60)+(float(ay[5:7]+'.'+ax[7:10])/3600)
)
                        print (long)
                    else:

long=(float(ay[0:2])+(float(ay[2:4])/60)+(float(ay[5:7]+'.'+ax[7:10])/360
0))
                        print (long)
                    es = Elasticsearch([{'host': IP_MAC, 'port':
PORT_MAC}], verify_certs=True)
#                       mapping = {
#                           "mappings":{
#                               "properties":{
#                               "timestamp": {"type":"date"},
#                               "coordenadas": {"type": "geo_point"},
#                               "IDC": {"type": "text"}
#                           }
#                           }
#                           }
#                       es.indices.create(index='geoloc', body=mapping)
                    es.index(index="prueba" ,body = {"timestamp":
datetime.utcnow(), "coordenadas": [long, lat], "IDC": str(IDC)})
                return 1
        else:
            print('GPS is not ready')
            return 0

def get_gps_position():
    rec_null = True
    answer = 0
    print('Start GPS session...')
    rec_buff = ''
    send_at('AT+CGPS=1,1','OK',1)
    time.sleep(2)
    while rec_null:
        answer = send_at('AT+CGPSINFO','+CGPSINFO: ',1)
        if 1 == answer:
            answer = 0
            if ',,,,,,' in rec_buff:
                print('GPS is not ready')
                rec_null = False
```

```
                time.sleep(1)
        else:
            print('error %d'%answer)
            rec_buff = ''
            send_at('AT+CGPS=0','OK',1)
            return False
        time.sleep(1.5)


def power_on(power_key):
    print('SIM7600X is starting:')
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(power_key,GPIO.OUT)
    time.sleep(0.1)
    GPIO.output(power_key,GPIO.HIGH)
    time.sleep(2)
    GPIO.output(power_key,GPIO.LOW)
    time.sleep(20)
    ser.flushInput()
    print('SIM7600X is ready')

def power_down(power_key):
    print('SIM7600X is loging off:')
    GPIO.output(power_key,GPIO.HIGH)
    time.sleep(3)
    GPIO.output(power_key,GPIO.LOW)
    time.sleep(18)
    print('Good bye')

try:
    power_on(power_key)
    get_gps_position()
    power_down(power_key)
except:
    if ser != None:
        ser.close()
    power_down(power_key)
    GPIO.cleanup()
if ser != None:
        ser.close()
        GPIO.cleanup()
```

Para recibir las medidas de los sensores y enviarlas por MQTT vía wifi a la Raspberry se ha utilizado el siguiente código Arduino en el nodeMCU:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#include "DHT.h"
#include "DHT_U.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#include <Wire.h>

//Direccion I2C de la IMU
#define MPU 0x68

//Ratios de conversion
#define A_R 16384.0 // 32768/2
#define G_R 131.0 // 32768/250

//Conversion de radianes a grados 180/PI
#define RAD_A_DEG = 57.295779
int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;

//Ángulos
float Acc[2];
float Gy[3];
float Angle[3];
float OldAngle[3];

int calca;
int calcb;
int calcc;
long marcador = 0;
long marcador2 = 0;

String valores;

long tiempo_prev;
float dt;


const char* ssid = "raspi-webgui";
const char* password = "ChangeMe";
const char* mqtt_server = "10.3.141.1";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
char msga[50];
char msgb[50];

int value = 0;
int alarma = 0;
void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
```

```
//  client.setCallback(callback);
  dht.begin();
Wire.begin(); //4,5 D2(GPIO4)=SDA / D1(GPIO5)=SCL
Wire.beginTransmission(MPU);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);


}

void setup_wifi() {

  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client")) {
      Serial.println("Conectado");
      client.publish("12345temperatura", "Enviando el primer mensaje");
      client.publish("12345humedad", "Empezando humedad");
      client.publish("12345alarma", "Empezando alarma");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void loop() {
```

```
  if (!client.connected()) {
    reconnect();
  }
  client.loop();




   //Leer los valores del Acelerometro de la IMU
   Wire.beginTransmission(MPU);
   Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
   Wire.endTransmission(false);
   Wire.requestFrom(MPU,6,true);   //A partir del 0x3B, se piden 6
registros
   AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
   AcY=Wire.read()<<8|Wire.read();
   AcZ=Wire.read()<<8|Wire.read();

   //A partir de los valores del acelerometro, se calculan los angulos Y,
X
   //respectivamente, con la formula de la tangente.
   Acc[1] = atan(-1*(AcX/A_R)/sqrt(pow((AcY/A_R),2) +
pow((AcZ/A_R),2)))*RAD_TO_DEG;
   Acc[0] = atan((AcY/A_R)/sqrt(pow((AcX/A_R),2) +
pow((AcZ/A_R),2)))*RAD_TO_DEG;
   Wire.beginTransmission(MPU);
   Wire.write(0x43);
   Wire.endTransmission(false);
   Wire.requestFrom(MPU,6,true);
   //Leer los valores del Giroscopio
   GyX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
   GyY=Wire.read()<<8|Wire.read();
   GyZ=Wire.read()<<8|Wire.read();

   //Calculo del angulo del Giroscopio
   Gy[0] = GyX/G_R;
   Gy[1] = GyY/G_R;
   Gy[2] = GyZ/G_R;

   //Aplicar el Filtro Complementario
   Angle[0] = 0.98 *(Angle[0]+Gy[0]*0.010) + 0.02*Acc[0];
   Angle[1] = 0.98 *(Angle[1]+Gy[1]*0.010) + 0.02*Acc[1];

   //Integración respecto del tiempo paras calcular el YAW
   Angle[2] = Angle[2]+Gy[2]*dt;

   //Mostrar los valores por consola
   //valores = "90, " +String(Angle[0]) + "," + String(Angle[1]) + "," +
String(Angle[2]) + ", -90";
   //Serial.println(valores);


  // Wait a few seconds between measurements.


  Serial.println(Angle [0]);
//
  Serial.print("\t\t");
  Serial.println(Angle [1]);
```

```
  Serial.print("\t\t");
  Serial.print(alarma);
  Serial.print("\t");
  //delay(10);
    if (millis() > (marcador2+10000)) {

    marcador2=millis();
    calca=abs(OldAngle[0]-Angle[0]);
    calcb=abs(OldAngle[1]-Angle[1]);
    calcc=abs(OldAngle[2]-Angle[2]);
    if (calca > 45 or calcb > 45 or calcc > 45)
      alarma=1;
    else
      alarma=0;
    OldAngle[0]=Angle[0];
    OldAngle[1]=Angle[1];
    OldAngle[2]=Angle[2];
    snprintf (msgb, 75,"%d", alarma);
    Serial.print("Alarm status: ");
    Serial.println(msgb);
    client.publish("12345Alarma", msgb);
    }


     if (millis() > (marcador+2500)) {
      marcador=millis();

    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
    //lastMsg = now;
    //++value;
    snprintf (msg, 75,"%.2f", temperature);
    Serial.print("Temperatura: ");
    Serial.println(msg);
    client.publish("12345temperatura", msg);
    snprintf (msga, 75,"%.2f ", humidity);
    Serial.print("Humedad: ");
    Serial.println(msga);
    client.publish("12345humedad", msga);

     }
}
```